Leitfaden zur Arbeit mit LaTeX, DVIPS und PostScript auf PCs und Unix-Workstations

Johannes Hoffstadt

Hochschulrechenzentrum Gießen

Vorwort

Das Textsatzsystem TEX wurde seit Ende der 70er Jahre von Prof. Donald Knuth entwickelt und erfreut sich nach wie vor großer Beliebtheit. Besonders oft wird LATEX von Leslie Lamport verwendet, um auf einfache Weise Dokumente zu erstellen. Früher oder später erreichen die meisten LATEX-Benutzer einen Punkt, an dem sie in ihren Dokumenten nicht mehr ohne Abbildungen auskommen. Die häufigste Lösung besteht darin, Platz im Text freizuhalten und dann mit Schere und Klebstoff zu hantieren. Das geht zwar schnell, aber es ist später umständlich, das Dokument in elektronischer Form weiterzugeben oder erneut zu verarbeiten. Ein anderer, oft geäußerter Wunsch ist der nach weiteren Schriften, sei es für fremdsprachliche Texte oder aus Geschmacksgründen. Man schaut sich um und stellt erfreut fest, daß es bereits sehr viele Schriften für TEX gibt. Doch sobald man von Ihnen Gebrauch machen will, tauchen unerwartete Probleme auf. Die neuen Schriften verunstalten Überschriften und Fußnoten, weil sie ihre ursprünglichen Attribute wie Größe und Form stur beibehalten. Und es erscheint erst recht aussichtslos, das ganze Dokument auf der Basis einer anderen Schriftfamilie als Computer Modern, der TEX-Einheitsschrift, zu setzen.

Viele geben sich an diesem Punkt damit zufrieden, so zurechtzukommen, wie sie es schon immer getan haben, obwohl es durchaus vernünftige Möglichkeiten gibt, diese Probleme zu lösen. Einer der möglichen Wege zum Erfolg läßt sich durch drei Stichworte beschreiben: **PostScript**, **dvips** und **NFSS**. Nur die ersten beiden tauchen im Titel auf, da NFSS bereits Bestandteil der neuesten LATEX-Version, LATEX 2_ε, ist.

Leider sind diese Möglichkeiten den meisten TEX-Anwendern zu diesem Zeitpunkt noch unbekannt, was nicht zuletzt an fehlender Literatur zu diesem Themenkomplex liegt. Ich versuche hiermit eine Lücke zu schließen und Ihnen, liebe Leserin, lieber Leser, vorzustellen, wie Sie den Anwendungsbereich von TEX erheblich erweitern können. Der Schwerpunkt dieses Leitfadens liegt auf den Gebieten *Grafik* und *Schriften*. Sie sollten Interesse an diesen zwei Themen mitbringen und Vorkenntnisse mit LATEX besitzen. Es spielt keine Rolle, ob Sie mit einem PC oder einer Unix-Workstation arbeiten. Ich persönlich arbeite mit Linux auf meinem PC. Auf vielen anderen Rechnern und Betriebssystemen können Sie ebenfalls mit TEX, dvips und PostScript arbeiten.

Der Leitfaden beginnt mit dem Thema PostScript. Sie werden erfahren, was PostScript ist und welche Aspekte davon für uns besonders nützlich sind. Anschließend beschreibe ich den PostScript-Interpreter **GhostScript**. Mit diesem Programm können Sie sich die grafischen Resultate von PostScript-Dateien auf dem Bildschirm oder auf verschiedenen Druckern darstellen lassen und so während den folgenden Kapiteln leicht mitarbeiten.

In Kapitel 3 geht es um **dvips**, das Bindeglied zwischen TEX und PostScript. Sie finden dort Hinweise zur Installation und zur Handhabung des Programms. Wir betrachten die prinzipiellen Fähigkeiten von

dvips (Schriften ausgenommen) und widmen uns in den nächsten zwei Kapiteln Diagrammen (mit **PSTricks**) und Grafiken (mit **epsf** und **epsfig**).

Die letzten zwei Kapitel behandeln Schriften. Eine Einführung in TEX-Schriften und NFSS, den neuen Schriftauswahl-Mechanismus für TEX, erhalten Sie in Kapitel 6. Dort finden Sie auch die Standard-PostScript-Schriften und den Umgang mit ihnen. Kapitel 7 ist für Experimentierfreudige gedacht. Sie werden erfahren, was „virtuelle Schriften“ sind, und wie Sie mit den Programmen **afm2tfm** und **vptovf** eigene virtuelle Schriften erzeugen können. Abgerundet wird die Theorie mit einem Beispiel, wie Sie eine PostScript-Schrift, die nicht zu dem Standardsortiment gehört, für TEX und GhostScript verfügbar machen können. Als Demonstrationsobjekt dient die Schrift *Utopia* von Adobe Inc.

Die Anhänge enthalten Informationen über eine einfache TEX-Installation unter Unix, viele Codetabellen, das PostScript-Namensschema von Karl Berry und ein Literaturverzeichnis.

Ich bedanke mich bei meinen freiwilligen „Lektoren“, Günter Partosch (Hochschulrechenzentrum der Justus-Liebig-Universität Gießen), Stefan Hofstetter (Institut für Theoretische Physik II, Universität Gießen), Roland Schmidt (Institut für Angewandte Physik, Universität Gießen) und Walter Obermiller (Max-Planck-Institut für Chemie in Mainz).

 Ich habe natürlich weiterhin ein offenes Ohr für Verbesserungsvorschläge, wenngleich ich nicht versprechen kann, sie sofort umzusetzen. Sie erreichen mich unter der email-Adresse

`johannes.hoffstadt@bio.uni-giessen.de`

Und nun viel Erfolg mit dieser Anleitung. Auf ans Ausprobieren!

Johannes Hoffstadt, im Mai 1994

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Kapitel 1

PostScript

1.1 Was ist eigentlich dieses PostScript?

Stellen Sie sich vor, Sie wollten einem Bekannten am Telefon ein Bild auf einem Blatt Papier beschreiben, das Sie vor Augen haben. Sie würden von Farben und Formen erzählen und von deren Lage auf dem Blatt. Ein Faxgerät würde dagegen Zeile für Zeile die Vorlage abtasten und nur die Wechsel von hell und dunkel übertragen. Ähnlich ist das Verhältnis von PostScript zu gewöhnlichen Druckersprachen.

PostScript wurde von der Firma Adobe Inc. entwickelt mit dem Ziel, eine hardwareunabhängige Seitenbeschreibungssprache zu schaffen. Diese Sprache enthält viele Funktionen für Grafik, Farbe und Schriften, bietet aber gleichzeitig alle Möglichkeiten einer gewöhnlichen Programmiersprache.

Die Grafikelemente von PostScript sind vektororientiert. Das bedeutet, es gibt Geraden, Kreisbögen und Kurvenzüge, die durch Koordinatenangaben (Vektoren) festgelegt sind, und Funktionen zum Skalieren, Verschieben und Drehen des aktuellen Koordinatensystems. Bei diesen Transformationen bleibt die Druckqualität erhalten.

Es gibt spezielle Drucker, die PostScript-Programme interpretieren können. Diese Drucker, meist Laserdrucker, setzen die Anweisungen um in ein Punkteraster mit einer bestimmten Punktdichte, dem *Auflösungsvermögen* des Druckers, z.B. 300 dpi (Punkte pro Zoll). Dieses Raster wird dann zu Papier gebracht. Je größer das Auflösungsvermögen ist, desto besser ist die Ausgabequalität.

Oft ist es günstig, vor dem Ausdrucken eine Vorabansicht der Druckseite auf dem Bildschirm zu bekommen. Dazu gibt es Programme, die ähnlich wie die in PostScript-Druckern eingebaute Software PostScript-Anweisungen interpretieren und auf das Punkteraster des Bildschirms zeichnen. Mit solchen Programmen können Sie oft auch Drucker ansteuern, die nicht von Hause aus PostScript-fähig sind.

Computergerechte Grafiken sind fast immer Rastergrafiken, die möglichst Punkt für Punkt, 1:1 auf dem Bildschirm abgebildet werden sollen. Ein Scanner zum Einlesen von Grafiken produziert

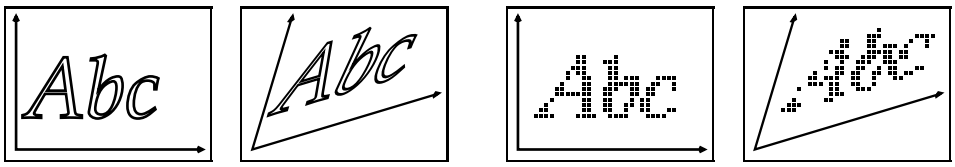


Abbildung 1.1: Die Qualität von Vektorgrafik bleibt bei Transformationen des Koordinatensystems erhalten. Rastergrafik leidet dagegen beträchtlich.

Dateien in verschiedenen Rasterformaten. Rastergrafik kann auch mit PostScript beschrieben werden. Eine Koordinatentransformation führt dabei zu unschönen Resultaten, da die Punkt-für-Punkt-Entsprechung aufgehoben wird. Rasterpunkte werden bei der Transformation verzerrt oder „fallen durchs Raster hindurch". Es entstehen Kanten und Störungen der Grafik (s. Abb. 1.1). In diesem Punkt ist die Wiedergabe von PostScript abhängig von der Auflösung des Ausgabegeräts.

Manche Workstations können bereits mit ihrer Systemsoftware PostScript am Bildschirm darstellen: **PageView** unter OpenWindows, **dxvdoc** auf der Alpha-Station unter OSF1/Motif oder **ShowPS** bei den neuesten AIX-Versionen. Voraussetzung dafür ist, daß „Display-PostScript" entweder die Grundlage der Bildschirmdarstellung ist (wie beim Betriebssystem NeXTstep) oder wenigstens als Extension im mitgelieferten X-Server integriert ist (z.B. bei DEC-Alpha, IBM-RS 6000 und Silicon Graphics).

1.2 Etwas Grundwissen über PostScript-Programme

PostScript-Programme bestehen aus lesbarem Text. „Binärzeichen", Sonderzeichen außerhalb des Standard-ASCII-Zeichensatzes (Codes 32–126), sind nicht akzeptabel, ausgenommen die Zeichen LF (Linefeed, 10) und CR (Carriage Return, 13), die einzeln oder gemeinsam ein Zeilenende bilden, das Tabulatorzeichen TAB (9) und das Zeichen EOT (End of Transmission, 4), das verwendet werden kann, um das Ende eines PostScript-Programmes zu markieren.

Es ist unerheblich, ob Ihre PostScript-Datei von einem DOS- oder einem Unix-System stammt; die verschiedenen Zeilenende-Zeichen werden richtig erkannt.[1] Manchmal allerdings werden Textdateien unter DOS mit dem Zeichen CTRL-Z (26) als Endemarke erzeugt. Das verträgt ein PostScript-Drucker nicht!

PostScript-Programme sind ansonsten formatfrei. Die einzelnen Wörter und Symbole werden durch Leerzeichen, TABs oder Zeilenenden getrennt. Es gibt in der Regel keine Begrenzung der Zeilenlänge. PostScript unterscheidet Groß- und Kleinschreibung.

[1] Wobei alte Versionen der DOS-Programme 'Freedom of Press' und 'GoScript' hier möglicherweise anderer Meinung sind.

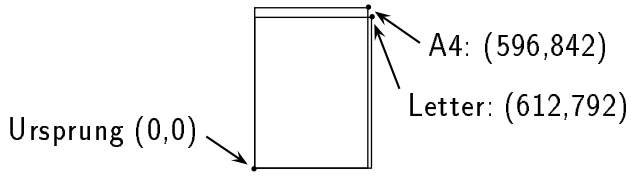


Abbildung 1.2: *Die Papierformate DIN-A4 und US Letter und ihre Größe in PostScript-Einheiten.*

Kommentare werden in PostScript durch ein Prozentzeichen eingeleitet. Von dort bis zum Zeilenende wird alles ignoriert außer dem Zeichen EOT.

An dieser Stelle ist ein Hinweis angebracht: Die folgenden Details sind zwar wissenswert, aber nicht essentiell wichtig für den Rest des Buches. Sie können den Abschnitt bis auf den letzten Absatz auf Seite 5 überspringen.

PostScript ist eine stackorientierte Sprache und hat etwas Ähnlichkeit mit der Programmiersprache Forth und der „umgekehrten polnischen Notation“, UPN, die oft von HP-Taschenrechnern benutzt wird. Das bedeutet vereinfacht gesprochen, daß alle Daten zuerst auf einem Stapel („Stack“) abgelegt werden, bevor „Operatoren“ mit den Daten arbeiten. Beispiel:

```
20 30 moveto
```

Zuerst werden die beiden Zahlen 20 und 30 auf dem Stapel abgelegt. Der Operator `moveto` holt sich die Koordinatenangabe vom Stapel und legt die aktuelle Zeichenposition, den sog. „current point“, fest.

Das (untransformierte) PostScript-Koordinatensystem benutzt als Einheit 1/72 Zoll[2] Der Ursprung des Koordinatensystems liegt an der Blattecke links unten. Wachsende x-Koordinaten gehen nach rechts, wachsende y-Koordinaten nach oben. In Abbildung 1.2 sehen Sie das DIN-A4- und das breitere, aber kürzere Letter-Format.

Es ist kein Problem, Seiten auf der Basis des Letterformats unverändert auf DIN-A4-Papier auszudrucken. Umgekehrt kann es passieren, daß Seiteninhalt, der zu weit oben steht, abgeschnitten wird.

PostScript kennt verschiedene Datentypen; unter anderem Zahlen, Namen und Zeichenketten. Zahlen erscheinen wie gewohnt. Namen werden durch einen Schrägstrich eingeleitet. Damit wird der Ausdruck wörtlich genommen, anstatt daß der PostScript-Interpreter versucht, ihn als Operator zu benutzen. Ein Beispiel dazu:

```
X
```

führt zu der Fehlermeldung „undefined“. Dagegen wird mit

```
/X 11 def
```

[2]Das ist etwas mehr als ein typographischer Punkt (72.27 pt=1 "). Die entsprechende Maßeinheit in TEX nennt sich daher 1 bp („big point“).

dem Namen 'X' die Bedeutung '11' zugewiesen. Und die wohlbekannte Konstruktion 'X=X+1' aus anderen Programmiersprachen schreibt sich in PostScript als

```
/X X 1 add def
```

Erst wird der nicht interpretierte Name 'X' auf den Stack gelegt. Beim zweiten Mal wird der zugeordnete Wert '11' abgelegt, gefolgt von der '1'. Der Operator `add` nimmt die beiden obersten Elemente vom Stack, addiert sie und legt das Ergebnis '12' wieder auf dem Stack ab. Der Operator `def` definiert den Namen 'X' neu als die Zahl '12'.

Man kann neue Operatoren als Abkürzungen für eine Gruppe von Anweisungen (Prozedur) definieren. Die Gruppierung erfolgt durch geschweifte Klammern:

```
/L { lineto } def
```

Hier wird eine Abkürzung für den `lineto`-Operator vereinbart. Jedesmal, wenn später ein `L` auftaucht, wird dafür `lineto` eingesetzt. Diese Konstruktion findet sich häufig am Anfang größerer PostScript-Dateien, da auf diese Weise viel Platz eingespart werden kann.

Manchmal ist es unerwünscht, daß für die neue Abkürzung immer wieder die Definition wörtlich eingesetzt wird, z.B. könnte sich die Definition von `lineto` ändern, ohne daß sich 'L' ändern soll. Man kann statt des einfachen wörtlichen Ersetzens die aktuelle *Funktion* von `lineto` mit dem neuen Operator `L` verbinden:

```
/L { lineto } bind def
```

Jetzt spielt es keine Rolle mehr, welche Bedeutung später dem `lineto`-Operator zugewiesen wird. `L` wird unabhängig davon immer eine Linie vom aktuellen Punkt zu den Koordinaten auf dem Stack ziehen.

Namen werden nicht nur gebraucht, um Abkürzungen zu definieren. Man verwendet sie auch für den Zugriff auf Schriften. Etwa so, zum Nachmachen:

```
/Times-Roman findfont 18 scalefont setfont
```

Der Operator `findfont` durchsucht die bekannten Schriften nach dem Namen 'Times-Roman', der in der Schreibweise exakt übereinstimmen muß. Die Schriftbeschreibung, normiert auf eine 1Punkt-Schrift, landet auf dem Stack. 18-fach vergrößert wird sie zur aktuellen Schrift ernannt.

Zeichenketten stehen in runden Klammern:

```
(Viel Spa\373) show
```

schreibt den Text „Viel Spaß“ in der aktuellen Schrift an die aktuelle Position. Innerhalb von Zeichenketten ist es manchmal notwendig, auf Sonderzeichen zuzugreifen. Das geschieht wie gezeigt über die Oktaldarstellung des Zeichencodes, die durch einen Backslash eingeleitet wird. Oft sind nicht alle Zeichen einer Schrift über einen Code erreichbar; das ist abhängig von der Codierung (Encoding) der Schrift. Das 'ß' hat in der *AdobeStandardEncoding* den Code 251, oder hexadezimal FB, oder oktal 373 (vgl. Tabelle B.3, dort werden Sie aber auch feststellen, daß die Umlaute nicht in der Codetabelle auftauchen).

Wichtig: Ein Operator, den Sie unbedingt kennen sollten, ist `showpage`. Dieser Operator veranlaßt PostScript-Drucker, die bisher als Raster im Druckerspeicher aufgebaute Seite nun zu Papier zu bringen, und danach eine neue, leere Seite zu beginnen. Fehlt `showpage`, so wird der Drucker die Datei schlucken, ohne eine Druckseite zu produzieren. Das ist bei EPS-Dateien oft der Fall.

1.3 PostScript-Schriften

Bei PostScript gibt es drei verschiedene Arten von Schriften, Type-1-, Type-3- und Type-5-Schriften. Alle sind skalierbare Vektorschriften.

Type-1-Schriften enthalten sogenannte *Hints* (könnte man mit „Tips“ übersetzen), mit denen die Schriften auch bei kleinen Größen noch gut aussehen. Type-1-Schriften sind codiert, aber das Verfahren ist vor wenigen Jahren veröffentlicht worden.[3]

Type-3-Schriften besitzen keine Hints. Dafür werden hier die einzelnen Zeichen über einen der Schrift eigenen Operator `BuildChar` ausgegeben, der im Prinzip jedes Zeichen individuell behandeln könnte. Diese Flexibilität haben die Type-1-Schriften nicht, denn sie verwenden immer dieselbe, fest im Interpreter eingebaute Type-1-`BuildChar`-Routine.

Type-5-Schriften sind Type-1-Schriften, die resident im Drucker vorliegen.

Nichtresidente Schriften müssen natürlich als Dateien existieren. Hier unterscheidet man zwei Varianten, `.pfa`- und `.pfb`-Dateien. Erstere sind lesbar („postscript font ascii“), die anderen binär codiert („binary“). Die binären Dateien sparen zwar viel Platz, können aber nicht direkt zum Drucker geschickt werden, da PostScript-Programme wie erwähnt nur ASCII-Zeichen enthalten dürfen. Dazu muß die erste Version verwendet werden, die alle Zeichen in Hexadezimalnotation überträgt.

PostScript-Drucker haben einen Standardvorrat von Type-5-Schriften. Meistens werden 35 Schriften mitgeliefert, seltener nur 13. Diese Schriften sind *lizenzpflichtig*, kosten also Geld (das im Preis eines PostScript-Druckers inbegriffen ist). Manche Type-1-Schriften sind im Rahmen der X11R5-Distribution zur Weitergabe freigegeben worden: Die Bitstream Charter Fonts, IBM Courier und Adobe Utopia sowie Varianten von URW Nimbus, Antiqua und Grotesk (s. Abbildung 1.3 auf Seite 6). Weitere Schriften sind als ShareWare erhältlich, etwa Kyrillisch oder die bekannten Hershey-Fonts.

[3] weil Adobe die erste ernsthafte Konkurrenz für PostScript durch das gemeinsame Kind von Microsoft und Apple, nämlich TrueType, fürchten mußte, und weil Hacker kurz zuvor ohnehin den Code geknackt hatten. Vor der Offenlegung des Codes mußten andere Firmen mit teurer Software von Adobe arbeiten, wenn sie auch Type-1-Fonts herausbringen wollten — und die sind nunmal die einzigen, die auch klein noch gut aussehen.

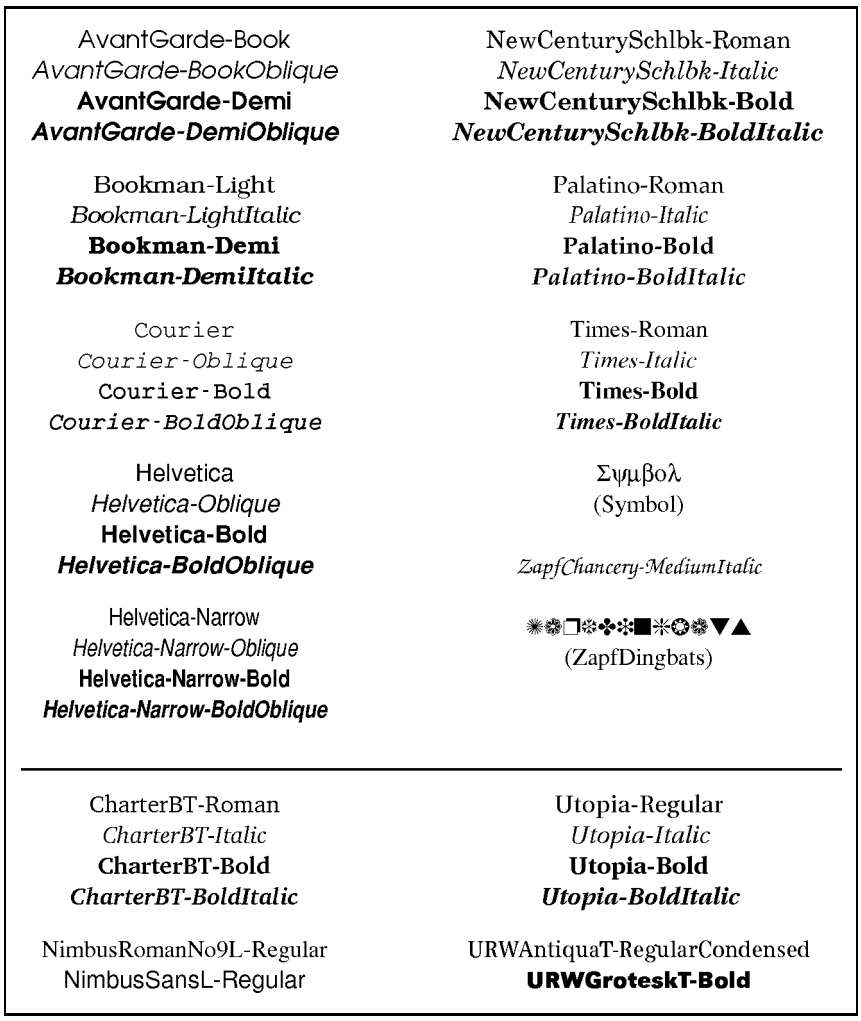


Abbildung 1.3: *Die 35 Standard-PostScript-Schriften (oben) und zusätzliche Type-1-Schriften aus der X11R5-Distribution (unten). — Der Minimalumfang der Standard-Schriften besteht aus den Familien Courier, Helvetica, Times und Symbol und umfaßt 13 Schriften.*

1.4 Encapsulated PostScript

Was ist „Encapsulated PostScript“ (EPS)? Wozu dient es? Wodurch unterscheiden sich EPS-Dateien von „normalen“ PostScript-Dateien? — Dazu ein Beispiel aus dem Bereich des Desktop Publishing: Die Arbeitsweise des Desktop Publishing gründet sich darauf, das Seitenlayout als Anordnung von rechteckigen Bausteinen anzusehen, die Grafiken oder Texte enthalten. Diese Grundidee sieht man jeder Zeitung an. In PostScript können solche Bausteine unabhängig voneinander beliebig zusammengesetzt werden, wenn sie der *Encapsulated-PostScript*-Spezifikation gehorchen. „Encapsulated“ kann man mit „gekapselt“ oder „in sich abgeschlossen“ übersetzen.

Der Name leitet sich daher ab, daß eine EPS-Datei den Rest des Dokuments nicht beeinflussen darf. Einige der „mächtigeren“ PostScript-Befehle sind in EPS-Dateien verboten.[4]

Die EPS-Datei muß der DTP-Software (oder TEX) mitteilen, welchen Platz die enthaltene Grafik beansprucht, damit entsprechender Raum freigehalten werden kann. Diese Art von Informationen über sich selbst liefern PostScript-Programme in speziellen Kommentarzeilen. Adobe hat für diesen Zweck eine Liste von Konventionen herausgegeben (die *Document Structuring Conventions*, DSC), die Auskunft über den Ursprung der Datei, die Anzahl der Seiten, die benötigten Schriften u.v.m. geben können. Alle solchen Kommentarzeilen beginnen mit einem doppelten Prozentzeichen am Zeilenanfang. PostScript-Dateien, die behaupten, diesen Konventionen zu genügen, beginnen mit der Sequenz

`%!PS-Adobe-`*PostScript-Version*

Für die Größe der Grafik ist der sogenannte `BoundingBox`-Kommentar zuständig. Die *BoundingBox* ist (im Idealfall) das kleinste Rechteck, das die Grafik einschließt. Angegeben werden die Koordinaten der linken unteren und der rechten oberen Ecke dieses Rechtecks. Für ein Objekt der Größe DIN-A4 würde dieser Kommentar so aussehen:

```
%%BoundingBox: 0 0 596 842
```

Beachten Sie die Schreibweise ohne Leerzeichen im Wort. Manche Programme ermitteln die BoundingBox zu einer EPS-Datei erst zum Schluß und verweisen mit

```
%%BoundingBox: (atend)
```

auf einen weiteren `BoundingBox`-Kommentar am Schluß der Datei. Wobei der Verweis am Anfang der Datei auch fehlen kann.

Es gibt ein PostScript-Programm namens `bb.ps`, das nachträglich fehlende `BoundingBox`-Kommentare ermitteln kann. Dazu definiert es sämtliche Zeichenoperatoren so um, daß diese vor ihrer eigentlichen Funktion immer wieder dieses umbeschriebene Rechteck aktualisieren. Schließlich gibt der modifizierte `showpage`-Operator zuerst die BoundingBox als Rechteck und als Text auf der Seite aus, bevor sie gedruckt wird. Für die Verwendung verweise ich auf Kapitel 2.6.

1.5 Warum TEX und PostScript?

Hier sind noch einmal die Eigenschaften aufgeführt, warum PostScript dazu prädestiniert ist, zusammen mit TEX verwendet zu werden:

- Mit PostScript erschließen sich dem TEX-Benutzer eine Reihe von professionellen Schriften, die in jeder beliebigen Größe eingesetzt werden können.

[4] Z.B. ist der Operator `showpage` aus verständlichen Gründen nicht erlaubt. Genausowenig darf das Koordinatensystem verändert werden. Es gibt ein Operatorenpaar speziell für den Zweck, die aktuellen Grafikeinstellungen zu sichern und zurückzuholen, die Befehle `gsave` und `grestore`. Der PostScript-Code in EPS-Dateien ist oft in ein `gsave`/`grestore`-Paar eingeschlossen.

- Mit PostScript können auf einfache Weise beliebige Abbildungen oder andere Layout-Bausteine in den Text eingebaut werden.

- Mit PostScript stehen etliche Grafik-Anweisungen zur Verfügung, die die in dieser Hinsicht bescheidenen Fähigkeiten von TEX erweitern.

- PostScript hält höchsten Qualitätsansprüchen stand. Sie können etwa einen Laserbelichter zur Herstellung von Offset-Druckfolien nutzen. Viele Druckereien nehmen bereits PostScript-Dateien als Repro-Vorlagen an.

1.6 Nachteile von PostScript? (Keine.)

Welche Nachteile bringt die Verwendung von PostScript mit sich? Erstens: Die Interpretation von PostScript verlangt leistungsfähige Hardware, sonst ist sie zu langsam, oder (bei zuwenig Speicher) sogar unmöglich. Zweitens: Bisher konnten Sie sich Ihr Dokument in einem speziellen Previewer für TEX ansehen. Diese Programme verstehen kein PostScript, daher bleiben alle PostScript-Spezialanweisungen dort wirkungslos. Erst mit einem PostScript-Interpreter für den Bildschirm können Sie Ihr Dokument begutachten. Ebenso wird für Drucker, die nicht PostScript-fähig sind, ein PostScript-Interpreter benötigt, wo vorher ein DVI-Treiber ausreichte.

Es gibt zwei gute Nachrichten dazu. Kostenlose PostScript-Interpreter sind für DOS, Unix und andere Systeme erhältlich. Wir werden von diesen das Programm **GhostScript** näher vorstellen. Außerdem ist die neueste Version des DVI-Treibers **xdvi** für Unix in der Lage, die Einbindung von PostScript-Bildern zu erkennen und auf Wunsch durch einen Aufruf von GhostScript im Hintergrund die fehlende Grafik nachzuliefern.

Schließlich gibt es kommerzielle PostScript-Interpreter. Für MS-DOS sind da z.B. die Programme **GoScript** oder **Freedom of Press** zu nennen, mit denen Sie PostScript-Dateien auf Ihrem Matrix-Nadeldrucker oder auf dem Bildschirm ausgeben können. Diese Programme sind auch in einer Version für MS-Windows erhältlich. Weitere Windows-Programme sind **ConvertPS**, **PowerScript** und **ZScript**.[5] Für Drucker werden spezielle PostScript-Module zum Einstecken angeboten, z.B. **Pacific-Page**-Module.

Fazit: Man kann mit PostScript leben, sofern die Hardware den Ansprüchen genügt; konkret: ab 386er aufwärts.

[5] Ein Vergleichstest der Zeitschrift c't kam allerdings zu der Erkenntnis, daß GhostScript alle kommerziellen Kandidaten in punkto Schnelligkeit, Stabilität und Leistungsfähigkeit schlägt. Die einzigen Nachteile von GhostScript sind Schriftausstattung und fehlende Treiberschnittstelle (d.h. es läßt sich nicht als Windows-Druckertreiber einsetzen.

1.7 Praxistips

Manchmal ist es sinnvoll, einen PostScript-Interpreter zu benutzen, obwohl PostScript-Drucker zur Verfügung stehen: Es kann in seltenen Fällen passieren, daß der Drucker die PostScript-Datei z.B. wegen Speichermangel nicht druckt. Oder Sie müssen ein komplexes Dokument mehrfach ausdrucken, und der Drucker braucht sehr lange für die Interpretation der Datei. Dann kann ein Interpreter helfen, der auf einem PC oder einer Workstation läuft, denn dort haben Sie meistens höhere Rechenleistung und mehr Hauptspeicher zur Verfügung. Einmal interpretieren, beliebig oft ausdrucken: Der Interpreter wandelt die PostScript-Datei z.B. in eine Druckdatei für einen HP Laserjet um, die darauf relativ schnell ausgedruckt werden kann.

PostScript-Dateien sind oft sehr groß, selbst wenn die langen Namen der Operatoren durch ein- oder zweibuchstabige Kürzel ersetzt werden. Eingebundene Bilder stehen beim Platzbedarf an erster Stelle, dann folgt die Anzahl und Größe der verwendeten nichtresidenten Schriften. Der Text selbst braucht wenig Platz. Abhilfe kann durch Datenkompression geschaffen werden: Da PostScript-Dateien aus lesbarem Text bestehen, sind sie besonders gut komprimierbar (nicht selten auf 15% der ursprünglichen Größe). Zu empfehlen sind die Archivierungsprogramme **pkzip**,[6] **lharc** oder **gzip**, die sowohl unter DOS als auch unter Unix laufen. Die gepackten Dateien sind sogar zwischen DOS und Unix problemlos austauschbar.

[6] pkzip ist allerdings nicht Public Domain, sondern Shareware, daher müssen Sie sich gegen Gebühr registrieren lassen, wenn Sie das Programm dauerhaft benutzen wollen.

Kapitel 2

GhostScript

Sie haben inzwischen vielleicht schon die ersten Versuche mit den neuen Möglichkeiten durch PostScript unternommen. Damit Sie sich PostScript-Dateien am Bildschirm ansehen können, folgt die Beschreibung von GhostScript. GhostScript ist ein *kostenloser* PostScript-Interpreter von Peter L. Deutsch (Aladdin Enterprises) u.v.a. Die aktuelle Version trägt die Nummer 2.6.1 vom März 1993. (Version 3.0 ist momentan — April 1994 — im Beta-Test-Stadium.)

2.1 Hinweise zur Installation

Installation unter Unix

Die Software-Installation ist unter Unix immer Sache des Systemadministrators. Hier beschreibe ich den Ablauf nur in groben Zügen, genaueres können Sie den mitgelieferten Hilfsdateien entnehmen.

Die Quelldateien für GhostScript sind als `tar`-Archiv erhältlich, meistens mit `gzip` gepackt. Zu der Version 2.6.1 kommen vier „Fixes“; außerdem sollten Sie GhostView (Version 1.5) installieren. Alles in allem brauchen Sie die Dateien

```
ghostscript-2.6.1.tar.gz
ghostscript-2.6.1fix-01.gz
ghostscript-2.6.1fix-02.gz
ghostscript-2.6.1fix-03.gz
ghostscript-2.6.1fix-04.gz
ghostscript-2.6.1-fonts.tar.gz
ghostview-1.5.tar.gz
```

Sie finden diese Dateien auf vielen FTP-Servern in deren `gnu`-Verzeichnis. Zum Beispiel auf

```
ftp.uni-stuttgart.de:pub/unix/gnu
```

Wählen Sie ein Arbeitsverzeichnis und packen Sie die erste Datei dort aus. Sie erzeugt ein Unterverzeichnis `gs261`. Wechseln Sie dort hin, entkomprimieren Sie die Patches und bringen Sie die Korrekturen an mit

```
cd gs261
patch -p < Patch-File
```

Dann kopieren Sie die passende Datei, `unix-cc.mak` oder `unix-gcc.mak`, nach `Makefile`. Lesen Sie die Gerätetreiberliste in `devs.mak` und tragen Sie alle gewünschten Treiber ins `Makefile` in die Zeile `DEVICE_DEFS=` ein. Jetzt können Sie versuchen, das Programm mit `make` zu übersetzen. Bei Problemen sollten Sie die beigefügten Texte lesen (`install.doc` usw.). Schließlich installieren Sie das Programm `gs`, die Shell-Skripte `gsbj`, `gsdj`, `gslj`, `gslp` und `gsnd` und die Hilfsdateien mit `make install`, das voreingestellte Ziel ist

```
/usr/local/bin  bzw.  /usr/local/lib/ghostscript
```

Nun können Sie das Schriftenarchiv ins Unterverzeichnis

```
/usr/local/lib/ghostscript/fonts
```

auspacken.

Bei der Installation von GhostView wird ein `Imakefile` verwendet. Zur Not können Sie auch mit `make` arbeiten, müssen dann aber das `Makefile` von Hand anpassen, was bei dem `Imakefile` größtenteils automatisch abläuft. Im Idealfall genügt die Eingabe der Befehle

```
xmkmf -a; make; make install
```

Installation unter MS-DOS

GhostScript für MS-DOS und MS-Windows sowie GhostView für MS-Windows sind als fertig compilierte Programme erhältlich, zu finden z.B. auf dem „neuen“ SIMTEL20-FTP-Server `oak.oakland.edu` oder einem seiner „Mirrors“, in Deutschland etwa

```
ftp.uni-heidelberg.de:/mirrors/msdos/postscrp
  gs261286.zip  bzw.  gs261386.zip  bzw.
  gs261win.zip  und  gsview10.zip  sowie
  gs261ini.zip
```

Packen Sie sie in ein eigenes Verzeichnis aus, etwa `C:\GS261`. Tragen Sie dieses Verzeichnis in den Suchpfad ein und setzen Sie die Umgebungsvariable `GS_LIB` auf das GhostScript-Verzeichnis. Das ist nötig, weil sehr wahrscheinlich ein anderer Pfad in das Programm eincompiliert wurde. Fügen Sie also z.B. die Zeilen

```
set GS_LIB=C:\GS261;C:\GS261\FONTS
set PATH=%PATH%;C:\GS261
```

ans Ende der Datei `AUTOEXEC.BAT` oder `SET-TEX.BAT` (von emTEX) an.

Die Schriften sind bei diesen Paketen nicht dabei. Nehmen Sie einfach die bei der Unix-Installation angegebenen Schriften und packen Sie sie in das Unterverzeichnis `GS261\FONTS` aus.

Bei Ghostview für Windows (Version 1.0) müssen die Dateien `GSVIEW.EXE`, `GSVIEW.HLP` mit in dem GhostScript-Verzeichnis stehen.

2.2 Bedienung von GhostScript

Sie rufen GhostScript mit dem Kommando `gs` auf, dem Sie einen oder mehrere Namen von PostScript-Dateien mitgeben können. Beispiel:

```
gs escher.ps
```

(`escher.ps` ist eine Demodatei von GhostScript. Weitere Demodateien sind z.B. `tiger.ps`, `golfer.ps` oder `snowflak.ps`.)

Unter DOS stoßen Sie hier auf das erste Problem: GhostScript schaltet in den Grafikmodus um und produziert die Seite auf dem Bildschirm, schreibt aber seine Meldungen zum Schluß mitten über die Grafik. — Abhilfe: Umleiten der Ausgabe irgendwohin. Beispiel:

```
gs escher.ps >gs.log
```

In der Datei `gs.log` finden Sie später etwaige Fehlermeldungen. Dieses Problem tritt unter Unix nicht auf, da GhostScript seine Meldungen in das Terminalfenster schreibt, in dem es aufgerufen wurde, für die Grafik dagegen ein eigenes X-Fenster öffnet.

Das zweite Problem: Die Seitendarstellung ist viel zu klein.[1]

— Abhilfe: Der Operator `scale` in PostScript skaliert das Koordinatensystem für die Ausgabe. Er braucht zwei Parameter für die x- und y-Vergrößerung, die PostScript-typisch vor dem Aufruf des Operators auf den Stapel gelegt werden. Also geben Sie innerhalb von GhostScript ein:

```
2 2 scale
```

Und ab jetzt ist alles doppelt so groß. Jetzt müssen Sie Ihre Datei erneut einlesen. Dateiname sind Zeichenketten, und Zeichenketten werden in PostScript in runde Klammern geschrieben. Der Operator `run` führt so angegebene Dateien aus. Also wieder innerhalb von GhostScript:

```
(escher.ps) run
```

Jetzt sehen Sie das Bild in doppelter Größe, und dabei gleich das dritte Problem. Vielmehr sehen Sie sie gerade nicht, die obere Hälfte des Bildes, die aufgrund der Skalierung jetzt nicht mehr auf den Bildschirm paßt. — Abhilfe: Das Koordinatensystem muß verschoben werden,

[1] Das ist natürlich eine Sache a) der Gewöhnung und b) der Grafikkarte. Mit dem ET4000-Treiber in der Auflösung 1024x768 kann man eine A4-Seite schon ganz gut lesen. Der entsprechende Aufruf für GhostScript lautet
`gs -sDEVICE=tseng -g1024x768` *PostScript-Datei*

das macht der Operator `translate`, und dann muß die Datei mal wieder neu gelesen werden. Beispiel, natürlich innerhalb von GhostScript:

```
0 -421 translate
```

verschiebt den Ursprung des Koordinatensystems um 421 PostScript-Einheiten, eine halbe DIN-A4-Seite, nach unten. Dadurch können Sie die obere Hälfte der Seite sehen. Die Rechts/Links-Verschiebung ist hier 0, aber ganz analog können Sie auch da Ihren Ausschnitt festlegen.

Das vierte Problem: Bei mehrseitigen Dokumenten (wie es mit TEX wohl der Fall sein wird) können Sie nur der Reihe nach durchblättern, ohne zwischendurch irgendwelche Kommandos wie die gerade erwähnten eingeben zu können. GhostScript wartet nach jeder Seite mit

```
>>showpage, press <return> to continue<<
```

darauf, daß Sie mit der Return-Taste weiterblättern. Wobei Sie diese Meldung nur ahnen, aber nicht sehen können, wenn Sie sie in die Datei `gs.log` umgelenkt haben.

2.3 Ghostview

Am Bildschirm gibt es nur eine sinnvolle Lösung, sowohl für Unix als auch für MS-Windows, jedoch nicht für MS-DOS: Sie brauchen einen Aufsatz namens **Ghostview**, über den GhostScript erst bequem bedienbar wird. Ghostview wurde von Timothy O. Theisen für X11 entwickelt, aber das Konzept wurde auch für ein MS-Windows-Programm benutzt, so daß die beiden Versionen im wesentlichen gleich zu bedienen sind.

Um GhostScript als Druckertreiber einzusetzen, ist kein Ghostview nötig, wenngleich es bequem ist, wenn man die zu druckenden Seiten einfach markieren kann, ohne sich mit PostScript-Kommandos herumzuschlagen.

Ghostview bietet Ihnen mehrere Menüs. Es gibt eine Liste mit Seitenzahlen des aktuellen Dokumentes, in der Sie einzelne Seiten gezielt anwählen können. Die Darstellung der Seite findet in einem rechteckigen Ausschnitt statt, dessen Lage auf der gesamten Seite durch zwei Schiebebalken verändert werden kann. Sie bekommen eine „gezoomte“ Darstellung durch Klicken auf eine Stelle innerhalb des Ausschnitts.

Normalerweise startet Ghostview zu Beginn einmal das Programm GhostScript, das dabei zuerst eine Initialisierungsphase durchläuft. Wenn Sie noch während einer laufenden Berechnung der Seitendarstellung die Seite wechseln o.ä., muß Ghostview den GhostScript-Prozeß abbrechen und neustarten. Mit der erneuten Initialisierung dauert das unter Umständen länger als wenn Sie einfach gewartet hätten, bis die Seite komplett berechnet war.

Wenn Sie in den Anzeigebereich hineinklicken, wird ein zweiter GhostScript-Prozeß gestartet und eine Ausschnittvergrößerung in einem separaten Fenster erstellt, das mit dem Knopf [Dismiss] wieder geschlossen werden kann. Jegliche Textausgaben und Meldungen des laufenden GhostScript-Prozesses erscheinen ebenfalls in einem neuen Fenster.

Das Menü File läßt Sie eine neue Datei laden, die alte erneut laden, die aktuelle Seite drucken, die markierten Seiten drucken oder abspeichern und schließlich das Programm verlassen. Bei Unix versucht Ghostview automatisch die alte Datei neu zu laden, wenn Ghostview vom Icon wieder zur Vollgröße vergrößert wird und die alte Datei inzwischen verändert wurde.

Das Menü Page läßt Sie blättern, die Seitendarstellung zentrieren und Seiten markieren und entmarkieren.

Im Menü Magstep stellen Sie die gewünschte Vergrößerungsstufe in festen Schritten von jeweils einem Faktor 1.2 (wie `magstep` in TEX) ein. `magstep 1` ist um den Faktor 1.2 größer, `magstep -1` um den gleichen Faktor kleiner als die Normalgröße, die von der Basisauflösung abhängt.

Das Menü Orientation bietet Ihnen die vier möglichen Lagen einer Textseite. Dabei hat ein in der PostScript-Datei angegebenes Seitenformat (als Bounding Box) zunächst Vorrang. Auf zwei Arten können Sie eine Änderung erzwingen, zum einen mit der Option '`-forceorientation`', zum anderen unter X durch Anwählen mit der mittleren statt mit der linken Maustaste. dvips versteht unter 'Landscape' etwas anderes als Ghostview; die Seite ist andersherum gedreht. Für solche Fälle gibt es die Kommandozeilenoption '`-swap`', die 'Landscape' und 'Seascape' vertauscht.

Im Menü Media wählen Sie aus amerikanischen, deutschen und anderen Papierformaten das gewünschte aus. Auch hier hat eine Angabe in der Datei Vorrang, Abhilfe schafft die Option '`-forcemedia`' bzw. wieder der mittlere Mausknopf. Um Ghostview mit dem Format A4 zu starten, verwenden Sie die Kommandozeilenoption '`-a4`'.

Es gibt unter Unix eine ganze Reihe weiterer Kommandozeilenoptionen (siehe Abbildung 2.1). Außerdem läßt sich das Unix-Ghostview über entsprechende X Resourcen konfigurieren. Sogar die englischen Menübeschriftungen können beliebig ersetzt werden. Systemweite Veränderungen sind in der Datei

```
/usr/lib/X11/app-defaults/Ghostview,
```

persönliche in

```
~/.Xdefaults
```

vorzunehmen. Fügen Sie etwa die Zeilen

```
Ghostview*pageMedia: A4
Ghostview*forcePageMedia: true
```

hinzu. Achtung, die Papierformate müssen hier im Gegensatz zu den Kommandozeilenoptionen groß geschrieben werden!

Diese Voreinstellungen beziehen sich nur auf den Start, sind also keineswegs permanent über die Laufzeit des Programms. Die Reihenfolge der Prioritäten ist: Kommandozeilenoptionen gehen vor lokalen Einstellungen, lokale vor systemweiten.

```
ghostview
   [-monochrome] [-grayscale] [-color]
   [-[no]title] [-[no]date] [-[no]locator] [-[no]labels]
   [-resolution <dpi>] [-dpi <dpi>]
   [-xdpi <dpi>] [-ydpi <dpi>] [-magstep <n>]
   [-[no]safer] [-[no]quiet] [-arguments <arguments>]
   [-[no]center]
   [-portrait] [-landscape] [-upsidedown] [-seascape] [-[no]swap]
   [-letter] [-tabloid] [-ledger] [-legal] [-statement]
   [-executive] [-a3] [-a4] [-a5] [-b4] [-b5]
   [-folio] [-quarto] [-10x14]
   [-force] [-forceorientation] [-forcemedia]
   [-[no]openwindows] [-[no]ncdwm]
   [-page <label>] [toolkitoption]
   [filename]
```

Abbildung 2.1: *Die Kommandozeilenoptionen von Ghostview*

2.4 Drucken mit GhostScript

Wenn Sie GhostScript mit der Option `-h` aufrufen, sehen Sie unter anderem auch die Liste der unterstützten Geräte (siehe Abbildung 2.2 und Tabelle 2.1). Sie sehen auch, wie Sie das Gerät auswählen, die Auflösung usw. setzen oder die Ausgabe in eine Datei lenken. Für die verbreitetesten Drucker gibt es vorgefertige Shell-Scripts bzw. Batch-Dateien `gsbj`, `gsdj`, `gslj` und `gslp` für Canon Bubble-Jet, HP Deskjet, HP Laserjet und Epson-Drucker. Nur ist dort in der Regel die voreingestellte Papiergröße unverändert. Für DIN A4 sollten Sie dort Änderungen anbringen. Ich betreibe meinen Canon BJ200 mit dem Aufruf

```
gs -sDEVICE=bj200 -sPAPERSIZE=a4
 -r360x360 -q -dNOPAUSE
 -sOutputFile=...
```

Beachten Sie die Groß- und Kleinschreibung!

2.5 GhostScript und Schriften

GhostScript kann im Gegensatz zu PostScript-Druckern mit beiden Dateiarten, `.pfa`- und `.pfb`-Dateien (s. Abschnitt 1.3), umgehen. Seine mitgelieferten Schriften haben die Endung `.gsf`, entsprechen aber durchweg `.pfa`-Dateien. Die Fonts befinden sich entweder im Installationsverzeichnis von GhostScript oder im Unterverzeichnis `fonts`.

GhostScript Device	Treiber für ...
appledmp	Apple Dot Matrix Printer, Imagewriter (nur Apple)
bj10e bj200	Canon Bubblejet
cdeskjet	HP Deskjet 500C 1Bit/Pixel
cdjcolor (=cdj500)	24 Bit/Pixel F-S Dithering
cdjmono	HP Deskjet 500C, schwarz
cdj550	HP Deskjet 550C
deskjet djet500	HP Deskjet, Deskjet Plus
djet500c	HP Deskjet 500C
declj250 lj250	DEC LJ 250 Companion Farbdrucker
dfaxhigh dfaxlow	DigiFAX Software Format
epson	EPSON-Drucker 9-/24-Nadel
eps9high	EPSON 9-Nadel 3fache Auflösung
epsonc	EPSON LQ-2550 und Fujitsu Farbdrucker
escp2	ESC/P 2 Kompatible, z.B. Stylus 800
ibmpro	IBM Proprinter 9-Nadel
jetp3852	IBM Jetprinter Farbtintenstrahldrucker
la50 la75, ln03	DEC LA50, LA75, LN03 Drucker
lbp8	Canon LBP-8 Laserdrucker
laserjet ljetplus	HP Laserjet / Laserjet Plus
ljet2p	HP Lj.IId/IIp/III* mit TIFF Komprimierung
ljet3	HP Lj.III* mit DeltaRow-Komprimierung
ljet4	HP Laserjet 4 (600 dpi)
m8510	C.Itoh M8510 Drucker
necp6	NEC Pinwriter (360x360 dpi)
nwp533	Sony NWP533 Laserdrucker (nur Sony)
oki182	Okidata Microline 182
paintjet	HP Paintjet Farbdrucker
pj pjxl	HP Paintjet XL (bis DIN A3)
pjxl300	HP Paintjet XL300
r4081	Ricoh 4081 Laserdrucker
sparc	SPARCprinter (nur SPARC)
t4693d2 t4693d4 t4693d8	Tektronix 4693d, 2/4/8 bits pro Farbkomp.
tek4696	Tektronix 4695/4696 Tintenstrahldrucker
trufax	TruFax FAX Treiber
bmpmono bmp16 bmp256	MS-Windows BMP Format
bmp16m gifmono gif8	Compuserve GIF Format
pcxmono pcxgray pcx16 pcx256	Paintbrush PCX Format
pbm pbmraw pgm pgmraw ppm ppmraw	Portable Bitmap/Graymap/Pixmap Format
tiffg3	TIFF Format

Tabelle 2.1: *Geräte- und Grafikformattreiber von GhostScript 2.6.1*

```
Ghostscript version 2.6.1 (5/28/93)
Copyright (C) 1990-1993 Aladdin Enterprises, Menlo Park, CA.
Usage: gs [switches] [file1.ps file2.ps ...]
Available devices:
   x11 bj10e bj200 cdeskjet cdjcolor cdjmono cdj500 cdj550
   declj250 deskjet dfaxhigh dfaxlow djet500 djet500c epson eps9high
   epsonc escp2 ibmpro jetp3852 laserjet la50 la75 lbp8
   ln03 lj250 ljet2p ljet3 ljetplus m8510 necp6 oki182
   paintjet pj pjxl pjxl300 r4081 t4693d2 t4693d4 t4693d8
   tek4696 bit bmpmono bmp16 bmp256 bmp16m gifmono gif8
   pcxmono pcx16 pcx256 pbm pbmraw pgm pgmraw ppm
   ppmraw tiffg3
Most frequently used switches: (you can use # in place of =)
   @<file>              treat file like part of the command line
                          (to get around DOS command line limit)
   -d<name>[=<token>]   define name as token, or null if no token given
begins with - or @
   -g<width>x<height>   set width and height ('geometry'), in pixels
   -I<prefix>           add prefix to search path
   -q                   'quiet' mode, suppress most messages
   -r<res>              set resolution, in pixels per inch
   -s<name>=<string>    define name as string
   -sDEVICE=<devname>   select initial device
   -sOutputFile=<file>  select output file: embed %d for page #,
                          - means stdout, use |command to pipe
'-' alone as a file name means read from stdin non-interactively.
For more complete information, please read the use.doc file.
```

Abbildung 2.2: *Die Kommandozeilenoptionen von GhostScript*

Da PostScript-Programme in der Regel die üblichen 35 Standardschriften voraussetzen, sind auch bei GhostScript entsprechende Schriften dabei. Wie schon erwähnt sind die Originalschriften *lizenzpflichtig* und dürfen deshalb nicht mit einem Paket wie GhostScript weitergegeben werden. Die Rechtslage ist aber in Bezug auf Schriften etwas eigenartig: Und zwar sind Vektorschriften (und damit die skalierbaren Originalschriften) durchaus als schützenswert anerkannt, aber Rasterschriften und Rasterbilder gelten — zu unserem Glück — offenbar als nicht copyright-fähig. Darum dürfen bei GhostScript gerasterte Versionen der Originalschriften beiliegen. Diese Rasterschriften sind jeweils für eine 24-pt-Schriftgröße erzeugt worden, allerdings ohne die Type-1-Hints, die gerade bei kleinen Schriften die Lesbarkeit verbessern würden.

Dabei sind leider stellenweise Fehler unterlaufen; bei der Ersatzschrift für Palatino fehlt mindestens das „ß“, es wird durch ein ineinandergeschobenes „ss“ ersetzt. Die beigefügte Ersatzschrift für ZapfDingbats kann von GhostScript gar nicht gelesen werden.

Was bei PostScript-Druckern die interne Liste der residenten Fonts ist, stellt die Datei `Fontmap` im GhostScript-Stammverzeichnis für GhostScript dar. Sie ist gewissermaßen das Gedächtnis von GhostScript, welche quasi-residenten Schriften es kennt. Wenn eine Schrift automatisch geladen werden soll, sobald ihr Name angegeben wird, muß die Schrift einen Eintrag in `Fontmap` besitzen. Die minimale Fontausstattung besteht aus dem Font *Ugly*. Wenn GhostScript eine angegebene Schrift nicht finden kann, wird sie als Notlösung durch *Ugly* ersetzt. Die Datei `Fontmap` enthält daher mindestens die Zeile

```
/Ugly               (uglyr.gsf);
```

Das Format eines Eintrags besteht also aus dem Namen der Schrift (mit führendem Schrägstrich) und dem Namen der Datei als Zeichenkette (in runden Klammern), in der die Schrift definiert ist. Die Zeile muß mit einem Semikolon enden.

Außer normalen Einträgen gibt es Alias-Einträge, die weitere Namen für vorhandene Schriften definieren. Zum Beispiel kennt GhostScript die Schrift *ZapfChancery-Oblique*, aber normalerweise wird diese Schrift unter dem Namen *ZapfChancery-MediumItalic* angesprochen. Dazu lauten die Einträge in der `Fontmap`:

```
/ZapfChancery-Oblique       (zcro.gsf);
/ZapfChancery-MediumItalic  /ZapfChancery-Oblique;
```

Sie können die Schriftdateien aber auch direkt in GhostScript laden wie jede andere PostScript-Datei auch. Wenn Sie z.B. die Schrift *Utopia-Regular* als `.pfa`-Datei haben, können Sie in GhostScript eingeben:

```
(Utopia-Regular.pfa) run
/Utopia-Regular findfont 24 scalefont setfont
```

2.6 Ermitteln von Bounding Boxes

Das schon beschriebene PostScript-Programm `bb.ps` dient dazu, die Bounding Box von PostScript-Dateien zu ermitteln, falls diese Angabe fehlt. Sie können die Bounding Box einer Datei ermitteln, indem Sie

`gs bb.ps` *Dateiname*

eingeben. `bb.ps` muß seine Umdefinitionen zuerst durchführen. Wenn die Datei nicht von sich aus mit `showpage` endet, erwartet GhostScript von Ihnen weitere Eingaben, und Sie können den fehlenden Befehl einfach eintippen. Es empfiehlt sich, auch eine modifizierte Version von `bb.ps` zu erzeugen, die die Zahlen nicht nur per `show` auf die Seite schreibt, sondern auch, oder stattdessen, mit `print` von GhostScript auf MS-DOS- bzw. Unix-Ebene zurückmeldet. Dann kann man diese Methode als Filter einsetzen, der automatisch in Dateien einen korrekten `BoundingBox`-Kommentar einträgt. Es ist sinnvoll, zu den Rändern sicherheitshalber 1–2 bp zuzugeben.

Kapitel 3

dvips

PostScript eröffnet in Zusammenarbeit mit TEX neue Perspektiven auf die Einbindung von Grafiken und die Nutzung von Schriften. Möglichkeiten, die über „das Normale“ in TEX hinausgehen, setzen jedoch voraus, daß der dvi-Treiber nicht nur die „normale“ dvi-Sprache übersetzen, sondern über Spezialkommandos all die wunderbaren Fähigkeiten von PostScript ausnutzen kann!

Tatsächlich kann TEX solche besonderen Nachrichten an den dvi-Treiber schicken. Das geschieht mit den `\special`-Befehlen, die Sie wahrscheinlich noch nie (wissentlich) benutzt haben. Es war ja bisher auch nicht sehr sinnvoll: Denn diese Befehle sind immer nur an einen ganz bestimmten dvi-Treiber gerichtet; alle anderen ignorieren sie. Und dann kann man seine `.dvi`-Datei zwar zuhause auf dem Nadeldrucker, aber nicht mehr im Büro auf dem Laserdrucker ausgeben.

Das emTEX-Paket bildet hier eine Ausnahme. Es enthält eine Auswahl von Druckertreibern, die alle eine bestimmte Sorte von `\special`-Befehlen verstehen, die sogenannten emTeX-Specials. Ein weiterer, inzwischen verbreiteter Satz von `\special`-Befehlen, die TPIC-Specials, dient zur Erweiterung der grafischen Fähigkeiten von LATEX in der `picture`-Umgebung.

dvips von Tomas Rokicki ist das gesuchte Programm, das nicht nur „normale“ `.dvi`-Dateien in PostScript umwandeln kann, sondern darüber hinaus über eigene `\special`-Befehle besondere PostScript-Funktionen zugänglich macht und zu guter Letzt auch noch emTeX- und TPIC-Specials versteht!

Mitgelieferte Style-Dateien erleichtern den Zugriff auf PostScript-Funktionen in TEX, so können Sie etwa eine TEX-Box (das, was man z.B. mit `\hbox{...}` erzeugt) mit einem einzigen Befehl drehen, uəddıʞ, ɹlǝʒǝiqs, usw. Das geht soweit, daß ein großes Grafik-Paket namens PSTricks (von Timothy van Zandt) speziell für die Verwendung mit dvips geschaffen wurde.

dvips ist farbtüchtig. Außer farbigem Text können Sie auch Grafiken, die Sie z.B. mit einem Malprogramm erzeugt oder mit einem Scanner eingelesen haben, sehr leicht in TEX-Dateien einbinden, was ein Schritt zur Lösung des leidigen Problems mit der Bebilderung von Texten ist. Details folgen im Kapitel 5 über Grafik.

Letzten Endes können Sie in Ihre TEX-Datei ganz beliebige PostScript-Befehle einfügen. Dazu sollten Sie sich mit PostScript auskennen (damit etwas Sinnvolles passiert). Die anderen

hier aufgeführten Möglichkeiten aber lassen sich sehr einfach auch ohne jegliche PostScript-Kenntnisse nutzen.

dvips kann auf Wunsch komprimierten PostScript-Code (allerdings nur für Schriften) erzeugen und so riesigen PostScript-Dateien entgegenwirken. Das Entkomprimieren muß dann aber der jeweilige PostScript-Interpreter durchführen, was u.U. ein wesentlicher Zeitfaktor sein kann. Reicht auch das Komprimieren nicht aus, um die Datei zu Transport- oder Archivierungszwecken auf eine Diskette zu bekommen, können Sie Ihr Dokument mit dvips in Abschnitte zu mehreren oder auch einzelnen Seiten aufteilen, also mehrere PostScript-Dateien erzeugen (wie das geht, im nächsten Kapitel) oder die überlange Datei auf mehrere Disketten verteilen.

3.1 Hinweise zur Installation

Installation unter Unix

Unter Unix ist die Installation von dvips eng mit der von TEX verbunden. Je nach Geschmack wählt man entweder die Originalpakete für TEX, dvips usw., die alle unabhängig voneinander zu installieren sind, oder die andere Möglichkeit, die ich persönlich vorziehe, ist das Komplettpaket, das von Karl Berry, dem Unix-Koordinator der TEX Users Group, regelmäßig überarbeitet und zur Verfügung gestellt wird. Die Installation des *gesamten* TEX-Systems ist damit so einfach, daß ich sie in Anhang A vorstellen möchte (auch wenn das nicht ganz zum Thema des Leitfadens paßt).

Die Standard-Distribution für dvips ist in der aktuellen Version die Datei

`dvips-5.55.tar.gz`,

zu finden z.B. auf dem FTP-Server

`ftp.uni-stuttgart.de:/pub/tex/dviware/dvips`

Packen Sie das Archiv aus, gehen Sie ins Verzeichnis `dvips` und lesen Sie die Installationsanleitung. Ihre Aufgabe ist es, in der Datei `config.h` und im `Makefile` die richtigen Pfade einzutragen und passende Einstellungen für dvips und für Ihr System anzugeben. Nur die Systemeinstellungen sind wirklich von Bedeutung, da sie darüber entscheiden, ob das Programm richtig compiliert werden kann. Falls Sie feststellen, daß die anderen Voreinstellungen unpassend gewählt waren, werden Sie gleich sehen, wie flexibel dvips umkonfiguriert werden kann (siehe Abschnitt 3.2).

Installation unter MS-DOS

Voraussetzung für die Arbeit mit dvips ist, daß Ihr PC mindestens einen 80286er-Prozessor hat. *Es ist sowohl aus Geschwindigkeitsgründen als auch prinzipiell vom Speicherbedarf der Programme her nicht sinnvoll, TEX und dvips auf einem 8086/8088-Prozessor betreiben zu wollen.* 1 MB

RAM ist das Minimum für dvips, und wenn Sie auch GhostScript nutzen wollen, brauchen Sie mehr Speicher. Normalerweise ist Ihr PC so konfiguriert, daß der gesamte eingebaute Speicher vollständig genutzt werden kann. Überprüfen Sie sicherheitshalber die Datei `CONFIG.SYS`: Am Anfang dieser Datei sollte durch die Zeile

`DEVICE=`*Pfad*`\HIMEM.SYS`

der „Erweiterungsspeicher“[1] zugänglich gemacht werden. Das gilt für ein MS-DOS-System; bei DR-DOS heißt das Programm `HIDOS.SYS`.

Wenn Sie einen 80286er-Rechner mit mehr als 1 MB Hauptspeicher besitzen, so kann dieser zusätzliche „Expansionsspeicher“[2] von Programmen nicht direkt benutzt werden. Dazu ist ein EMM-Speichertreiber erforderlich (expanded memory manager), der nach `HIMEM.SYS` in der Konfigurationsdatei `CONFIG.SYS` angegeben sein sollte.

Wenn Sie dagegen einen 80386er-, 80486er-, ...-Rechner haben, wird der Speicher oberhalb von 1 MB über das Emulatorprogramm `EMM386.EXE` als Expansionsspeicher behandelt. Dieses Programm ist bei MS-DOS und bei MS-Windows im Lieferumfang enthalten. In diesem Fall sollte nach dem Eintrag `HIMEM.SYS` die folgende Zeile erscheinen:

`DEVICE=`*Pfad*`\EMM386.EXE RAM` (ab MS-DOS 5.0)

Dabei geben Sie an, wieviel von Ihrem Hauptspeicher oberhalb von 1 MB zu EM-Speicher werden soll.

Wenn Ihnen diese Kurzinformation nicht genügt, empfehle ich Ihnen, Ihr MS-DOS- oder MS-Windows-Benutzerhandbuch zu Rate zu ziehen.

Auf der Software-Seite bestehen keine besonderen Voraussetzungen, dvips arbeitet mit jeder ordnungsgemäßen TEX-Implementation für PCs zusammen. Sie finden das fertig compilierte Programm ebenso wie GhostScript auf den SIMTEL-Mirrors, etwa

```
ftp.uni-heidelberg.de:
    /mirrors/msdos/postscrp/dvips554.zip
```

Anschließend müssen Sie eine neue Umgebungsvariable einrichten: Die Variable `TEXCONFIG` muß den Pfad enthalten, unter dem die Datei `config.ps` zu finden ist. Bei emTEX-Installationen ist das normalerweise `\EMTEX\PS`. Editieren Sie also die Datei `SET-TEX.BAT` oder die Datei `AUTOEXEC.BAT` und hängen Sie die folgende Zeile an:

`SET TEXCONFIG =` *Laufwerk*`:\EMTEX\PS`

Dabei müssen Sie das Laufwerk einsetzen, auf dem sich Ihre gesamte emTEX-Installation befindet.

[1] extended memory, XM: Speicherbereich oberhalb des konventionellen Arbeitsspeichers mit Speicheradressen von 640 KB bis 1 MB

[2] expanded memory, EM: Speicher oberhalb von 1 MB in einem 80286-Rechner.

3.2 Konfiguration von dvips

Die Datei `config.ps` enthält Pfade und andere Einstellungen für dvips. Es gibt mehrere Möglichkeiten, um zwischen verschiedenen Sätzen von Einstellungen, z.B. für verschiedene Drucker, auszuwählen. dvips sucht zuerst im aktuellen Verzeichnis nach einer Konfigurationsdatei, somit haben persönliche Einstellungen Vorrang vor den systemweiten. Falls das erfolglos bleibt, wird mit Hilfe der Umgebungsvariable `TEXCONFIG` die systemweite Konfigurationsdatei gelesen. Wenn die Variable nicht gesetzt ist, werden Voreinstellungen verwendet, die beim Compilieren von dvips benutzt wurden. Auf Workstations muß dvips ohnehin neu compiliert werden; bei dieser Gelegenheit werden normalerweise die richtigen Pfade eingetragen, so daß man auf die Variable verzichten kann. Dagegen wird für PCs ein fertig compiliertes dvips angeboten, dessen Voreinstellungen für Sie aller Wahrscheinlichkeit nach unbrauchbar sind. Auf PCs sollten Sie die Variable also wie oben beschrieben setzen.

Wenn Sie verschiedene Einstellungen für dvips brauchen, etwa weil Sie zwei unterschiedliche Drucker betreiben möchten, *könnten* Sie entsprechende `config.ps`-Dateien in *verschiedene* Verzeichnisse legen und mit der Variable `TEXCONFIG` auswählen. Aber es ist übersichtlicher, die Dateien in einem einzigen Verzeichnis zu halten. Dazu müssen sie unterschiedlich heißen. Sie können dvips über die Option '`-P`' einen anderen Suffix als `.ps` vorgeben; z.B. um zwischen `config.lj` (für einen Laserjet) und `config.p6h` (für einen NEC Pinwriter) wählen zu können. Darauf kommen wir bei der Besprechung der Optionen noch einmal zurück.

Nachdem dvips versucht hat, eine lokale oder systemweite Konfigurationsdatei zu lesen, gibt es unter Unix noch eine letzte Auswahlmöglichkeit: Wenn Sie eine Datei namens `.dvipsrc` in Ihrem HOME-Verzeichnis stehen haben, wird diese zum Schluß noch gelesen, wobei schon vorher getroffene Einstellungen noch einmal verändert werden können. Höchste Priorität haben aber immer die Optionen, die Sie in der Kommandozeile angeben.

Jetzt haben wir in einiger Länge den Initialisierungsprozeß von dvips besprochen, aber noch nicht kennengelernt, was alles eingestellt werden kann. Neben einigen Pfaden sind das z.B. die Basisauflösung des Druckers in dpi und der Modus für die Schriftberechnung durch METAFONT, der einem optimalen Parametersatz für das Druckwerk des Druckers entspricht. Ein Eintrag besteht aus einem Kennbuchstaben und dem zu setzenden Wert. Kommentarzeilen beginnen mit '*'; Leerzeilen sind erlaubt. Die Reihenfolge der Einträge spielt keine Rolle.

Für gewöhnlich sieht die Datei `config.ps` für emTEX etwa so aus:

```
M hplaser
D 300
T D:\EMTEX\TFM
V D:\TEXFONTS\VF
P D:\TEXFONTS\PIXEL.LJ\%dDPI\%f.%p
L D:\FONTLIBS\;HPLASER
S .;D:\EMTEX\TEXINPUT
H .;D:\EMTEX\PS
...
```

Das bedeutet, daß METAFONT den Modus „hplaser“ benutzen soll, die Basisauflösung beträgt 300 dpi; es folgen Pfadangaben für .tfm-, .vf- und .pk-Dateien (T, V, P), für Fontlibraries (L),[3] für einzubindende Dateien wie Bilder und für die PostScript-Header-Files (H). Die Pfad-Angaben beziehen sich auf das Laufwerk D:. Wenn Ihr TEX auf einem anderen Laufwerk installiert ist, müssen Sie diese Datei editieren.

Die Kombinationen „Prozentzeichen+Buchstabe“ in der Zeile, die mit 'P' beginnt, sind Platzhalter. Es gibt '%d' für die benötigte Auflösung, '%f' für den Schriftnamen und '%p' für den Typ der Schriftdatei (heutzutage immer '.pk', früher war auch '.pxl' gebräuchlich).

Angenommen, dvips suche die Schrift 'cmr17' in der Vergrößerungsstufe[4] zwei bei der obigen Voreinstellung. Mit der 'P'-Zeile ergibt sich der folgende vollständige Pfad:

```
D:\TEXFONTS\PIXEL.LJ\432DPI\CMR17.PK
```

Bei einer TEX-Installation auf Workstations sind andere Pfade voreingestellt. Wir haben z.B. Unterschiede in folgenden Einträgen:

```
T .:/usr/local/lib/tex/fonts
V .:/usr/local/lib/tex/fonts/vf
P .:/usr/local/lib/tex/fonts/pk
*L .:/usr/local/lib/tex/fontlibs
S .:/usr/local/lib/tex/inputs
H .:/usr/local/lib/tex/ps
```

Beachten Sie, daß Verzeichnisse bei Unix-TEX nicht durch ein Semikolon, sondern durch Doppelpunkte getrennt werden. Die Schriftdateien befinden sich hier alle in einem Verzeichnis. Die 'P'-Zeile ist jetzt eine Kurzform für

```
P .:/usr/local/lib/tex/fonts/pk/%f.%d%p
```

Das Format '%f.%d%p' ist nämlich die Voreinstellung für die Bildung der Schriftdateinamen. In unserem Beispiel entsteht der Pfad

```
/usr/local/lib/tex/fonts/pk/cmr17.432pk
```

Was Fontlibraries angeht, gilt: Da die Verwaltungsprogramme für die Pflege der Fontlibraries nur auf dem PC existieren, wird man sich überlegen, ob man sie überhaupt auf Workstations einsetzen will. Das ist eigentlich nur sinnvoll, wenn Sie die Hauptarbeit mit TEX auf dem PC erledigen, und es genügt, ab und zu aktualisierte Fontlibraries auf die Workstation hochzuladen. Dann muß die mit '*L' beginnende Zeile auskommentiert werden (der Stern muß entfernt

[3] Die zu emTEX gehörenden dvi-Treiber und auch dvips können nicht nur einzelne .pk-Dateien verarbeiten, sondern auch mit Schriftbibliotheken (Font-Libraries) umgehen. Die üblichen zig .pk-Dateien liegen dann nicht einzeln auf der Platte vor, sondern sind in einer großen Datei gebündelt. Dabei spart man Plattenplatz und hat wesentlich schnelleren Zugriff auf die Schriften.

[4] TEX verwendet für Schriften eine abgestufte Vergrößerungsreihe (*magsteps*), bei die Größe von Stufe zu Stufe um den Faktor 1.2 zunimmt. Ausgehend von der Basisgröße bedeutet z.B. \magstep 2 einen Faktor von 1.44. Es gibt zusätzlich einen halben Schritt, genannt \magstephalf, von $\sqrt{1.2} \approx 1.095$.

werden). Üblicherweise zieht man eine von PCs unabhängige Fontverwaltung vor und verzichtet auf Fontlibraries.

Nun gesetzt den Fall, Sie benötigen für Ihren Drucker Fonts mit 360 dpi Auflösung statt mit 300 dpi (z.B. für NEC Pinwriter oder Canon Tintenstrahldrucker). Dann können Sie auf dem PC die Fontlibrary `NECP6HI.FLI` verwenden und diese in die Datei `config.ps` eintragen. Setzen Sie dann auch die Basisauflösung auf 360, bei Nadeldruckern den METAFONT-Modus auf `LQHIRES` und den Pfad zu den `.pk`-Dateien auf das Verzeichnis `PIXEL.P6H`. Für einen Tintenstrahldrucker ist der Modus `LQHIRES` allerdings ungeeignet, da die Tinte deutlich kleinere Punkte erlaubt als eine Nadel, die ein Farbband aufs Papier drückt. Entsprechend werden ganz andere Werte für den Schwärzungsgrad (`blacker`) und andere Parameter (`fillin`, `o_correction`) benötigt. Tintenstrahldrucker und Laserdrucker[5] sind sich m.E. ähnlich genug, daß man den Modus `HPLASER` auch für Tinte verwenden kann.

Was tun, wenn Sie schon alles erdenkliche ausprobiert haben, aber dvips immer noch Schriftdateien nicht findet oder sonst aus unerfindlichen Gründen abbricht? Dann sollten Sie mit der Option '`-d64`' mitverfolgen, welche Dateien es zu öffnen versucht. So sehen Sie leicht, wo der Irrtum liegt.

Es gibt noch weitere Einstellungen, die Sie in der Konfigurationsdatei treffen können. Dort können Sie das Papierformat auswählen und häufig gebrauchte Optionen aus der Kommandozeile festlegen. Aber das würde hier zu sehr ins Detail führen. Lesen Sie bei Bedarf die Originalanleitung von dvips.

3.3 Der Aufruf von dvips

Im einfachsten und häufigsten Fall haben Sie eine `.dvi`-Datei und möchten diese „ohne Sonderwünsche“ in eine PostScript-Datei umwandeln. Dazu geben Sie das Kommando

`dvips` *Dateiname*

ein und schon meldet sich dvips und beginnt, eine Datei mit dem Namen der `.dvi`-Datei, aber der Endung `.ps` zu erzeugen, in der sich dann der Inhalt Ihrer `.dvi`-Datei nach PostScript übersetzt befindet. Diese PostScript-Datei können Sie dann zum Drucker schicken.

Dieses indirekte Verfahren hat einen kleinen Nachteil: Sie müssen auf dem Speichermedium, auf dem Sie diese Übersetzungsaktion ausführen, noch *genügend Speicherplatz* für die PostScript-Datei haben. Wenn das nicht der Fall ist, müssen Sie direkt an den Drucker ausgeben (kommt gleich).

Natürlich können Sie beim Aufruf von dvips auch noch einige Parameter angeben. Sie bekommen eine gute Übersicht, wenn sie einfach nur

```
dvips
```

[5] Genauer gesagt, Laserdrucker mit Write-Black-Technik.

```
This is dvipsk 5.526a Copyright 1986, 1993 Radical Eye Software
     Usage: dvips [options] filename[.dvi]
 a*  Conserve memory, not time      y # Multiply by dvi magnification
 b # Page copies, for posters e.g.  A   Print only odd (TeX) pages
 c # Uncollated copies              B   Print only even (TeX) pages
 d # Debugging                      C # Collated copies
 e # Maxdrift value                 D # Resolution
 f*  Run as filter                  E*  Try to create EPSF
 h f Add header file                F*  Send control-D at end
 i*  Separate file per section      K*  Pull comments from inclusions
 k*  Print crop marks               M*  Don't make fonts
 l # Last page                      N*  No structured comments
 m*  Manual feed                    O c Set/change paper offset
 n # Maximum number of pages        P s Load config.$ s
 o f Output file                    R   Run securely
 p # First page                     S # Max section size in pages
 q*  Run quietly                    T c Specify desired page size
 r*  Reverse order of pages         U*  Disable string param trick
 s*  Enclose output in save/restore V*  Send downloadable PS fonts as PK
 t s Paper format                   X # Horizontal resolution
 x # Override dvi magnification     Y # Vertical resolution
                                    Z*  Compress bitmap fonts
     # = number   f = file   s = string  * = suffix, '0' to turn off
     c = comma-separated dimension pair (e.g., 3.2in,-32.1cm)
```

Abbildung 3.1: *Kommandozeilenoptionen von dvips*

eingeben (siehe Abbildung 3.1). Detaillierte Erklärungen der Optionen finden Sie in der englischen Original-dvips-Anleitung.

Die Optionen werden durch Buchstaben bezeichnet. So steht etwa '`-o`' für 'output', '`-c`' für 'copies' usw. *Bei den Optionen wird zwischen Groß- und Kleinschreibung unterschieden!* So können Sie die Ausgabe durch '`-o`' umlenken, nicht aber durch '`-O`'.

Sie können den von dvips erzeugten PostScript-Dateien einen anderen Namen als den der `.dvi`-Datei geben. Wenn Ihre `.dvi`-Datei `extra.dvi` heißt, sie aber möchten, daß die erzeugte PostScript-Datei nicht `extra.ps`, sondern `normal.ps` genannt wird, geben Sie das Kommando

```
dvips -o normal.ps extra
```

ein. (Wie Sie sehen, können Sie bei der `.dvi`-Datei die Endung auch weglassen.) Wenn Sie z.B. die Ausgaben von dvips direkt auf einen angeschlossenen PostScript-Drucker ausgeben wollen, so wählen Sie

unter MS-DOS: `dvips -o prn` *Dateiname*
oder `dvips -o lpt1:` *Dateiname*
unter Unix: `dvips -o !lpr`*Dateiname* ,

falls der Drucker am LPT1:-Anschluß des PCs hängt bzw. der Unix-Befehl `lpr` die Warteschlange des PostScript-Druckers anspricht (geregelt durch die Umgebungsvariable `LPDEST`).

Wenn Sie jede Seite in dreifacher Ausfertigung haben möchten, so geht das ganz einfach:

`dvips -c 3` *Dateiname*

erzeugt eine PostScript-Datei, durch die jede Seite dreimal hintereinander gedruckt wird. Sie können nach '`-c`' eine beliebige Zahl zwischen 1 und 255 angeben. Wenn Sie aber die ganze Datei dreimal hintereinander ausgeben möchten, um den Ausdruck hinterher nicht auseinandersortieren zu müssen, geben Sie ein

`dvips -C 3` *Dateiname*

Allerdings wird die PostScript-Datei dadurch dreimal so lang. Die Buchstaben '`C`' und '`c`' stehen für 'Collated copies' bzw. 'uncollated copies'.

Die Optionen '`-l`' und '`-p`' ('last page' bzw. 'first page') dienen dazu, die Datei nur teilweise auszugeben:

`dvips -l 12` *Dateiname*	bis Seite 12
`dvips -p 3` *Dateiname*	ab Seite 3
`dvips -p 3 -l 12` *Dateiname*	die Seiten 3–12

Damit hat man eine Möglichkeit, zu groß geratene PostScript-Dateien wieder in den Griff zu bekommen: Übersetzen Sie Ihre `.dvi`-Datei einfach seiten- oder abschnittsweise nach PostScript! Etwa:

```
dvips -l 12 -o teil1.ps Dateiname
dvips -p 13 -l 24 -o teil2.ps Dateiname
dvips -p 25 -o teil3.ps Dateiname
```

Die entstehenden Fragmente lassen sich dann leichter handhaben. Eine Batch-Datei, die für jede Seite eine einzelne PostScript-Datei erzeugt, ist schnell geschrieben. — Übrigens sind die so angegebenen Seitennummern genau die Nummern im TEX-Register `\count0` (also die, die auf der Seite auch erscheinen). Wenn Sie dagegen die 'wahren' Seitennummern brauchen, setzen Sie '=' vor die Nummer.

Alternativ dazu können Sie die Schriften im PostScript-Code komprimieren lassen (Option '`-Z`'). Das kostet allerdings Zeit, da der PostScript-Drucker die Schriften vor Gebrauch wieder entpacken muß.

Jetzt komme ich nochmal zu der Auswahl von verschiedenen Druckern: Angenommen, Sie haben sich für jeden Drucker eine Kennzeichnung überlegt und eine Datei `config.`*Kennzeichen* eingerichtet wie oben erwähnt. Dann wählen Sie mit der Option `-P` den Drucker aus, also

`dvips -P lj` *Dateiname* ,

um die Konfigurationsdatei `config.lj`, die für einen Laserjet gedacht sein könnte, auszuwählen.

Falls es Ihnen einmal passiert, daß beim Ausdruck einer PostScript-Datei die letzte Seite „verschluckt“ wird, dann kann es sein, daß der Drucker das Ende des Druckauftrages nicht erkennt

(er möchte dann eine Endemarke (*EOT*) haben). Handelt es sich um eine PostScript-Datei unbekannter Herkunft, müssen Sie sich damit behelfen, die Endemarke von Hand eingeben, indem Sie die Datei editieren und ganz am Schluß ein `CTRL-D` anfügen. Mit dvips können Sie auch die Option '`-F`' verwenden, die genau dasselbe bewirkt. Um sicherzugehen, können Sie diese Option jedesmal setzen, das schadet nicht und verlängert Ihre PostScript-Datei nur um ein Byte.

dvips kann, wenn nötig, über die Option '`-O`' (wie Offset) die Druckseite verschieben (ab Version 5.5). Darauf folgt ein Wertepaar durch Komma getrennt; der erste Wert gibt die Verschiebung nach rechts, der zweite nach unten an. Wenn die ausgegebene Datei zu weit links oben sitzt, so können Sie die Ausgabe manipulieren, ohne daß Sie etwas an der TEX-Datei ändern müssen. Das ist insbesondere bei PostScript-Druckern nützlich, die ein falsches DIN-A4-Format zugrundelegen oder DIN A4 gar nicht kennen. In so einem Falle ist es einfacher, die Ausgabe zu verschieben, da jede Änderung der TEX-Datei nur Katastrophen auf allen anderen Ausgabegeräten produzieren würde.

Angenommen, Sie möchten die Ausgabe Ihrer `.dvi`-Datei um einen Zentimeter nach links und um zwei Zentimeter nach oben verschieben. 1 cm nach links entspricht -1 cm nach rechts, dann lautet der Aufruf:

```
dvips -O -1cm,2cm Dateiname
```

Das war eigentlich schon alles. Mehr brauchen Sie nicht zu wissen, wenn Sie einfach nur TEX-Dateien am PostScript-Drucker ausdrucken möchten.

3.4 „Specials“

Bis jetzt haben wir nichts kennengelernt, was nicht auch andere dvi-Treiber bieten. Mit den `\special`-Befehlen stoßen wir nun ins Reich der Möglichkeiten mit PostScript vor. Wir wollen uns nur auf die Befehle beschränken, die dvips zu eigen sind, *emTeX*- und *TPIC*-Specials besprechen wir nicht.

Einer dieser `\special`-Befehle dient zum Festlegen des Papierformats. Nun läßt sich das auch über die Kommandozeile von dvips erledigen, aber sinnvollerweise schreibt man das Papierformat (als einen Teil des Dokumentenentwurfs) doch besser in die TEX-Datei hinein. Querformat stellt man etwa mit

```
\special{landscape}
```

ein. Oder wenn Sie ganz bestimmte Blattabmessungen haben, sagen Sie

```
\special{papersize=14.85cm,21cm}
```

Dabei geben Sie erst die Breite, dann die Höhe an (14.85 cm$\times$21 cm entspricht DIN A5). *Beachten Sie: dvips schaut dann in seiner Konfigurationsdatei nach, ob dort ein passender Eintrag steht, wenn nicht, nimmt es den erstbesten Eintrag und ignoriert Ihre* `\special`*-Anweisung.*

Das ist auch sinnvoll, denn es geht hier um reale Blattformate, nicht um den bedruckten Bereich. Wenn man eine Papiergröße wählt, die der Drucker (laut Konfigurationsdatei) nicht liefern kann, so nimmt dvips eben das Format, das verfügbar ist. — Der Landscape-Modus ist aber bei jedem definierten Papierformat möglich, Sie dürfen nur nicht vergessen, auch die Textabmessungen in TEX entsprechend zu setzen.

Ein weiteres Spezialkommando dient dazu, am Anfang der von dvips erzeugten PostScript-Datei einige vorbereitende Definitionen einzusetzen. Es lautet

```
\special{header=Dateiname}
```

dvips selbst schickt eine Reihe eigener Definitionen voraus (die Dateien `tex.pro`, `texc.pro`), von denen das nachfolgende PostScript-Programm ausgiebig Gebrauch macht. Gewisse „virtuelle Fonts“ brauchen einen Codierungsvektor wie `ec.enc` als Header-Datei, was Sie später im Kapitel über PostScript-Schriften nachlesen können. Damit haben Sie aber nichts weiter zu tun, auch wenn Sie solche Schriften verwenden, denn dvips erkennt anhand seiner Tabelle der PostScript-Schriften (die Datei `psfonts.map`) automatisch, ob es entsprechende Header-Dateien in seine PostScript-Ausgabe einbauen muß. Auch das Paket PSTricks definiert sich eigene Befehle in der Datei `pstricks.pro`, wobei sich die Angelegenheit für Sie darauf beschränkt, 'pstricks' als Stiloption am Anfang des Dokuments anzugeben.

Es genügt demnach, wenn Sie vage wissen, daß es solche Header-Dateien gibt (Sie sehen alle verwendeten Header-Dateien zu Beginn eines dvips-Durchlaufs aufgeführt), es sei denn, Sie kennen sich mit der Programmierung in PostScript aus und planen, eigene Style-Dateien, etwa für Logos, zu entwerfen. Dann wird Sie der nächste Absatz interessieren.

Sie können eigene PostScript-Befehle direkt in die TEX-Datei einbauen. Eingeleitet wird dieses wörtliche PostScript im Befehl `\special` durch ein Anführungszeichen[6] und könnte etwa so aussehen:

```
\special{"newpath 0 -5 moveto 8 -5 lineto
8 -60 lineto 0 -60 lineto closepath gsave
0.9 setgray fill grestore stroke}
```

So eingebundener PostScript-Code wird von dvips in ein `gsave`/`grestore`-Paar eingeschlossen und ist für manche Anwendungen zu eingeschränkt. Es gibt noch eine höhere Stufe der Kontrolle: Sie geben statt des Anführungszeichens `ps:` an. Damit können Sie ohne Einschränkungen alles in PostScript mögliche tun — falls Sie wissen, was Sie da tun.

Beispiele für die Anwendung finden Sie im `rotate`-Style, der zu dvips gehört, oder im `outline`-Style, den ich vor einiger Zeit mal geschrieben habe: Damit können Sie PostScript-Schriften als Umriß darstellen (auf unglaublich ineffektive Weise, aber es geht meistens).

Ein weiteres nützliches Kommando ist

```
\special{psfile="Dateiname"}
```

[6] Achtung `german.sty`-Benutzer! Dort ist das Anführungszeichen ein aktives Befehlszeichen! Abhilfe: `\mdqoff` schaltet diese Aktivität ab, `\mdqon` wieder an.

Damit werden beliebige PostScript-Dateien an der aktuellen Position ins Dokument eingefügt. Auch die Makros zum Einbinden von Bildern verwenden diesen `\special`-Befehl.[7]

Gerade bei Abbildungen tritt oft das schon erwähnte Problem auf, daß die PostScript-Dateien dazu sehr groß werden können. Gleichzeitig sind Sie aber auch gut komprimierbar. dvips läßt Sie daher statt der riesengroßen Datei auch ein Kommando angeben, das eine gepackte Datei dekomprimiert und auf der Standardausgabe ausgibt. Dieses Kommando wird durch einen „backtick“ eingeleitet. Unter Unix läßt sich das wie folgt nutzen:

```
\epsffile[Bounding Box]{"`zcat riesig.ps.Z"}
```

So wird die Datei `riesig.ps`, die komprimiert vorliegt, in Ihr Dokument eingebunden. Beachten Sie aber, daß TEX im diesem Fall keine Möglichkeit hat, die Bounding Box aus der Datei zu lesen, so daß Sie sie von Hand angeben müssen. Außerdem sollte Ihnen klar sein, daß die von dvips erzeugte PostScript-Datei bereits das entkomprimierte Bild enthält und entsprechend groß sein kann. Doch das war ein Vorgriff auf Kapitel 5, in dem die Einbindung von Bildern detailliert beschrieben ist.

Sie können nach der Konstruktion "` nicht nur `zcat`, sondern jedes beliebige Unix-Kommando ausführen lassen, vorausgesetzt, es ist im Suchpfad zu finden; ob es sinnvoll ist, dessen Ausgabe in der PostScript-Datei stehen zu haben, müssen Sie selber wissen. Unter DOS funktioniert der Aufruf eines Kommandos nicht — wenn Sie Ihre Dateien portabel lassen wollen, müssen Sie auf den „backtick“ verzichten.

Schließlich führt dvips vier besondere PostScript-Operatoren aus, wenn Sie sie definieren, nämlich `start-hook` und `end-hook` am Anfang und am Ende des Dokuments, sowie `bop-hook` und `eop-hook` zu Beginn und am Ende jeder Seite.

Als Beispiel soll die Konstruktion dienen, mit der dieser Text auf einem zweiseitigen PostScript-Drucker in DIN A5 quer so ausgegeben wird, daß der Ausdruck nur in der Mitte durchgeschnitten und geheftet werden muß, um ein Buch zu erhalten. Zuerst muß die Seitengröße in TEX auf A5 gesetzt werden. Der linke Teil von Abbildung 3.2 zeigt die Wirkung: Das Seitenbild muß offensichtlich noch gedreht werden. Das schon erwähnte Kommando

```
\special{landscape}
```

dreht zwar, aber nur die geraden Seiten liegen vom Rand her richtig, wie Sie in der Mitte der Abbildung sehen können. Die ungeraden Seiten müssen noch um 180° gedreht werden. Dazu eignet sich der Operator `bop-hook`, den ich so definiert habe:

```
\special{!userdict begin /bop-hook{
   2 index 2 mod 1 eq{595.2 1262.7
   translate -1 -1 scale}if }def
```

Der Mechanismus läuft nun folgendermaßen: Wenn der Operator `bop-hook` definiert ist, wird er zum Beginn jeder Seite aufgerufen, wobei die aktuelle TEX-Seitenzahl und die „wahre“ Seitenzahl auf dem Stack liegt (und am Ende der Prozedur immer noch dort liegen muß). Mit ‘2

[7] Neue Versionen von xdvi kennen diesen Befehl und rufen GhostScript auf, um das Bild im Preview anzeigen zu können.

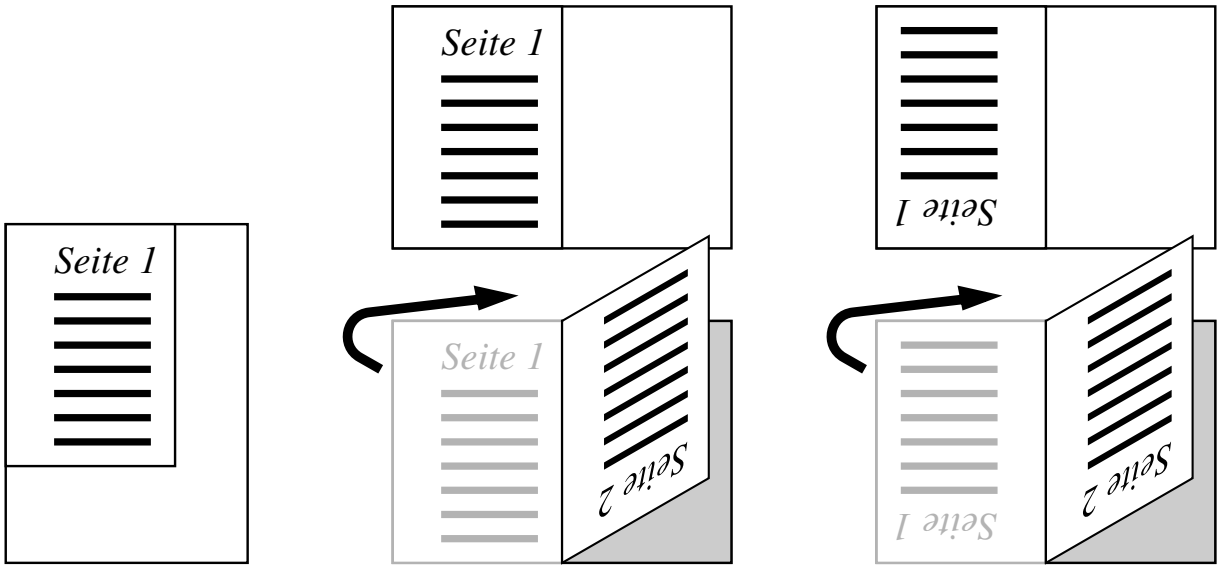


Abbildung 3.2: *Zusammenwirken des* `landscape`*-Specials und des Operators* `bop-hook`*, um auf zweiseitigen PostScript-Druckern im A5-Format auszugeben. Links: Ausgabe von dvips, wenn das A5-Format in TEX gewählt wird; mitte: Wirkung des* `landscape`*-Specials — bei doppelseitigem Druck sind nur die geraden Seiten richtig ausgerichtet; rechts: durch passende Definition des Operators* `bop-hook` *(s. Text) werden auch die ungeraden Seiten richtigherum gedruckt.*

`index`' wird die „wahre" Seitennummer oben auf den Stapel kopiert. '`2 mod`' bildet den Divisionsrest mit 2. Damit liegt entweder '0' oder '1' auf dem Stack, je nachdem, ob die Seitenzahl gerade oder ungerade ist. Ist sie ungerade ('`1 eq {...} if`'), wird der Seiteninhalt umgedreht, ist sie gerade, bleibt er unverändert. Das Ergebnis ist in der rechten Hälfte der Abbildung 3.2 zu sehen. Jetzt passen ungerade und gerade Seiten beim Umblättern zusammen. Nach einem Ausdruck kann der Papierstapel umgedreht und auf der anderen Hälfte bedruckt werden. Nach dem Durchschneiden in der Mitte hat man zwei bindegerechte Exemplare.

Soviel zu den `\special`-Kommandos. Diese Möglichkeiten sind nicht unbedingt geeignet, um sie jedesmal von Hand zu nutzen, aber auf ihnen beruhen viele Style-Dateien, wie etwa `epsfig` oder `pstricks`, die ein komfortableres Bedienen erlauben. `epsfig` wird im Kapitel über Grafikeinbindung erklärt, und zu PSTricks kommen wir jetzt.

Kapitel 4

PSTricks

Sie haben vielleicht einmal mit der `picture`-Umgebung in LaTeX oder deren Erweiterung PicTeX versucht, Diagramme zu erstellen. Dann haben Sie sicher die vielen damit verbundenen Einschränkungen in guter Erinnerung behalten: Die `picture`-Umgebung kann nur wenig, sie kann nicht mal beliebig geneigte Linien zeichen, und mit PiCTeX stößt man allzu leicht an Speichergrenzen, denn da werden die Linien mangels Schriften aus Einzelpunkten aufgebaut. Aus sehr vielen Punkten! Jeder braucht als TeX-Box genauso viel Speicher wie ein Buchstabe. Stellen Sie sich mal eine Seite mit 100000 Buchstaben vor: Dafür ist TeX nun wirklich nicht gedacht.

PSTricks ist von der Handhabung her ähnlich wie die `picture`-Umgebung, was man natürlich nicht gerade als komfortabel bezeichnen kann. Aber mit PSTricks hat man immerhin die ganze Fülle der PostScript-Möglichkeiten zur Verfügung, und die Bindung an TeX-Boxen besteht nicht mehr, denn TeX bekommt von den Specials gar nichts mit. PSTricks ist sogar kompatibel zu plainTeX und $\mathcal{AMS}$TeX, nicht nur zu LaTeX. Die allgemeine Akzeptanz der PSTricks kann man auch daran messen, daß fremde Programme wie **gnuplot** unter anderem auch TeX-Dateien mit PSTricks-Anweisungen erzeugen können (ab Version 3.0).

Der größte Nachteil bei der Verwendung von PSTricks besteht darin, daß man beim Entwurf seiner Grafiken — oder sollte man *Programmierung* sagen? — oft einige Kontrolläufe braucht, bis das Ergebnis paßt, und da wirkt sich die Schwachstelle der Kombination TeX-PostScript negativ aus. Sie können Ihr Werk nicht im normalen Previewer sehen, also müssen Sie nach dem LaTeX-Durchlauf einen dvips-Durchlauf anschließen und dann einen PostScript-Treiber bemühen, der auch nicht zu den schnellsten Programmen zählt. Sie kommen schneller zum Ziel, wenn Sie Ihre Diagramme jeweils in einer separaten, kleinen Datei entwerfen.

4.1 Allgemeine Vorinformationen

Bei PSTricks ist eine sehr schöne Anleitung (auf Englisch) dabei, die detailliert die verschiedenen Befehle demonstiert. Schon beim Blättern gewinnt man einen Eindruck der Möglichkeiten und findet jeweils neben dem Ergebnis als Bild die TeX-Zeilen, mit denen das Bild erzeugt werden

kann. Ich kann hier aus Platzgründen keine umfassende Anleitung geben, sondern nur ein paar Appetithappen, damit Sie sich leichter dazu aufraffen können, sich die Originaldokumentation mal näher anzuschauen.

Als erstes müssen Sie die Datei `pstricks.sty` in die Stiloptionen der Anweisung

```
\documentstyle[...,pstricks,...]{...}
```

aufnehmen bzw. die Zeile

```
\usepackage{pstricks}
```

hinzufügen, falls Sie mit LaTeX 2_ε arbeiten. Es gibt noch einige andere Stiloptionen, die zu PSTricks gehören und zusätzliche Funktionen bereitstellen (`pst-node`, `multido`, `gradient`, ...). In der PSTricks-Anleitung ist jeweils vermerkt, welche Stiloptionen Sie zusätzlich benötigen.

Folgende Konventionen gelten für die Argumente von PSTricks-Kommandos: Argumente stehen in geschweiften {...}, optionale Argumente in eckigen Klammern [...], Koordinaten (Zahlenpaare) in runden Klammern (x,y). Zusätzliche Variablen und Parameter werden mit einem Gleichheitszeichen gesetzt, bestehen also aus *Schlüsselwort=Wert*-Paaren. Mehrere solche Angaben werden durch Kommata getrennt.

PSTricks definiert übrigens gleich ein paar Grundfarben als einfache TeX-Kommandos, z.B. \gray, \red, \green, \blue, \yellow, \white.

Das sehen Sie natürlich nur richtig auf einem Farbausdruck oder z.B. mit GhostScript.

4.2 Grafikparameter

Bei PSTricks läßt sich fast alles einstellen, von den aktuellen Koordinateneinheiten über die Linienbreite bis zur Zeichenfarbe. Diese Parameter können individuell beim Kommandoaufruf in eckigen Klammern angegeben werden — das sieht dann etwa so aus:

```
\psline[linewidth=2mm](0,0)(1,1)
```

— oder global voreingestellt werden, was für alle nachfolgenden Kommandos gilt. Dazu dient das Kommando `\psset`:

```
\psset{linewidth=2mm}
```

In Tabelle 4.1 sehen Sie einige dieser Parameter, ihre Voreinstellungen, und welche Kommandos dadurch beeinflußt werden. Die `picture`-Umgebung besitzt den Parameter `\unitlength`, der die Einheit für alle Koordinatenangaben festlegt. Bei PSTricks können Sie gleich zwei Skalierungsfaktoren setzen, nämlich `xunit` und `yunit`, etwa

```
\psset{xunit=1cm}
\psset{yunit=1cm}
```

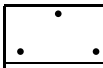
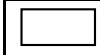
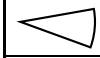
	`\psdots(0,0)(0.5,0.5)(1,0)`
	`\psline(0,0)(0.5,0.5)(1,0)`
	`\pspolygon(0,0)(0.5,0.5)(1,0)`
	`\psframe(0,0)(1,0.5)`
	`\pscircle(0.5,0.25){0.25}`
	`\psellipse(0.5,0.25)(0.5,0.25)`
	`\pswedge(0,0.3){1}{-20}{10}`
	`\pscurve(0,0)(0.5,0.4)(0.2,0.3)(0.6,0.5)(1,0)`

Abbildung 4.1: *Grundlegende Grafikbefehle von PSTricks*

Anders als in der `picture`-Umgebung können Sie auch vollständige Maßangaben (d.h. mit Einheit) unabhängig von den eingestellten Einheiten angeben!

4.3 Einige Grafikelemente

Einige grundlegende Kommandos sind in Abbildung 4.1 gezeigt. Befehle, die Grafikelemente wie Linien, Ellipsen, Rechtecke oder Kreissegmente zeichnen, gibt es auch in *gesternten* Versionen, die das Objekt ausfüllen. Aber nicht in der gesetzten `fillcolor`, wie man vermuten könnte, sondern in der `linecolor`!

Linien können viel mehr als einfache Verbindungslinien sein. Sie können die Linienbreite vorgeben und auch doppelte, gestrichelte und punktierte Linien zeichnen lassen. Sie können runde Kreise, Pfeile oder Doppelpfeile, T-Streifen usw. an den Endpunkten haben. Die Eckpunkte können abgerundet werden. Dabei haben Sie die Wahl, ob die Linienenden genau die Endkoordinaten treffen sollen oder ob Sie noch um die halbe Liniendicke als Radius herausragen sollen. Sie finden einige Varianten in Abbildung 4.2.

Ich hatte schon die gesternten Befehlsvarianten erwähnt, die die Figur ausfüllen. Sie können auch bei den ungesternten Befehlen zusätzlich zu `linecolor` und `linestyle` die Parameter `fillcolor` und `fillstyle` angeben. Damit sind Schraffuren ein Kinderspiel; Änderungen der Linienbreite, des Abstands, der Farben, der Neigung erfolgen über Grafikparameter. Insgesamt gesehen erschlägt einen die Vielfalt der Optionen. Selbst die Form der Pfeilspitzen kann durch Parameter variiert werden!

Dann gibt es da die Kurven, Polygonzüge und Plot-Funktionen. Mit dem Zusatz-Style `pst-plot` können Sie Werte aus Dateien plotten lassen und schöne Koordinatensysteme mit beschrifteten

Parameter	Wertebereich	Voreinstellung	Bemerkungen
Koordinaten...			
xunit	*Maß*	1 cm	für x-Koordinaten
yunit	*Maß*	1 cm	für y-Koordinaten
runit	*Maß*	1 cm	für alle anderen
origin	{*Maß*, *Maß*}	{0 pt, 0 pt}	verschiebt Ursprung
swapaxes	true/false	false	tauscht Achsen aus
Linien und Kurvenzüge...			
linewidth	*Maß*	0.8 pt	
linecolor	*Farbe*	black	
linestyle	*Stilangabe*[a]	solid	
dash	*zwei Zahlen*	5pt 3pt	Muster für dashed
dotsep	*Maß*	3 pt	Abstand für dotted
border	*Maß*	0 pt	Breite einer Umrandung
bordercolor	*Farbe*	white	und deren Farbe
doubleline	true/false	false	Doppelte Linien?
doublesep	*Maß*	1.25 linewidth	...und deren Abstand
doublecolor	*Farbe*	white	Farbe des Zwischenraums
shadow	true/false	false	Schatten?
shadowsize	*Maß*	3 pt	Größe
shadowangle	*Winkel*	-45	Richtung
shadowcolor	*Farbe*	darkgray	Farbe
dimen	outer/middle/ inner	outer	\psframe, \pscircle, \psellipse, \pswedge
linearc	*Maß*	0 pt	\psline, \pspolygon
framearc	*Zahl* (0--1)	0	\psframe und Verwandte
Ausfüllen...			
fillstyle	*Stilangabe*[b]	none	
fillcolor	Farbname	black	
hatchwidth	*Maß*	0.8 pt	Linienbreite
hatchsep	*Maß*	4 pt	Schraffurabstand
hatchcolor	*Farbe*	black	
hatchangle	*Winkel*	45	
showpoints	Wahrheitswert	false	zeigt Kontrollpunkte
dotstyle	Punktcode[c]	*	\psdots, showpoints
dotscale	zwei Zahlen	1 1	horiz./vert. Skalierung
dotangle	Winkel (Grad)	0	rotiert Punktdarstellung
curvature	drei Zahlen	1 0.1 0	\pscurve und Verwandte
arrows	*Stilangabe*[d]	-	Linien aller Art

[a] Mögliche Werte für linestyle sind: none, solid, dashed, dotted.

[b] Werte: none, solid, vlines, vlines*, hlines, hlines*, crosshatch, crosshatch*. Gesternte Varianten füllen den Hintergrund aus (wie solid).

[c] Werte: *, o, +, triangle, triangle*, square, square*, pentagon, pentagon*, |.

[d] Werte: -, <->, >-<, <<->>, >>-<<, |-|, |*-|*, [-], (-), o-o, *-*, oo-oo, **-**, c-c, cc-cc, C-C.
Die linken und rechten Enden dürfen nach Belieben miteinander kombiniert werden. Die leeren und ausgefüllten Kreise sitzen zentriert auf den Endpunkten (bei o, *, c) oder schließen bündig ab (oo, **, cc).

Tabelle 4.1: *Eine Auswahl der Grafikparameter von PSTricks*

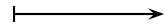
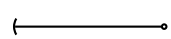
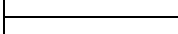
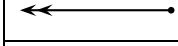
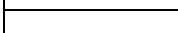
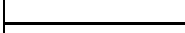
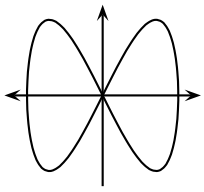
	`\psline[arrows=|->](0,0)(2,0)`
	`\psline{(-o}(0,0)(2,0)`
	`\psline{<<-*}(0,0)(2,0)`
	`\psline[linestyle=dashed](0,0)(2,0)`
	`\psline[linestyle=dotted](0,0)(2,0)`
	`\psline[doubleline=line](0,0)(2,0)`
	`\psline[linewidth=2mm](0,0)(2,0)`
	`\psline[linewidth=2mm]{C-C}(0,0)(2,0)`
	`\psline[linewidth=2mm]{c-c}(0,0)(2,0)`
	`\psline[linewidth=2mm]{cc-cc}(0,0)(2,0)`

Abbildung 4.2: *Verschiedene Arten von Linien. Der Parameter* `arrows` *kann statt in eckigen Klammern (erstes Beispiel) auch kürzer in geschweiften Klammern angegeben werden.*



```
\parametricplot[plotstyle=ccurve]%
{0}{360}{t sin t 2 mul sin}
\psline{->}(0,-1.2)(0,1.2)
\psline{<->}(-1.3,0)(1.3,0)
```

Abbildung 4.3: *Eine Lissajous-Figur als Beispiel eines parametrischen Plots.*

Achsen erzeugen. Mit ein paar PostScript-Kenntnissen können Sie auch Funktionen oder parametrische Plots darstellen. Abbildung 4.3 zeigt eine Lissajous-Figur ($x = \sin t$, $y = \sin 2t$) aus der Original-Anleitung.

4.4 Höhere Grafikbefehle

Die PSTricks-Anleitung zeigt, wie Sie sich eigene komplexe Grafik-Objekte (*custom graphics*) definieren können, wobei Sie nah an die teilweise gefährlichen Innereien von PostScript gelangen (die Anleitung unterscheidet hier „safe tricks“, „pretty safe tricks“ und „for hackers only“). Beachten Sie aber bitte, daß es Verdruß bringen kann, wenn Sie durch unsicheren PostScript-Code einen Netzwerkdrucker zum Absturz bringen! Es ist dagegen völlig risikolos und obendrein

papiersparend, sich vorher mit GhostScript ein Bild Ihres Bildes zu verschaffen.

PSTricks stellt Ihnen einen Ersatz für die LaTeX-`picture`-Umgebung bereit, der sich `pspicture` nennt. Sie geben auch hier die Abmessungen des Bildes an, damit PSTricks genug Platz reservieren kann (als Box). Wie bei `picture` kann das Bild außerhalb des angegebenen Rahmens liegen. Wenn Sie aber `pspicture*` verwenden, wird außerhalb des Rahmens abgeschnitten (clipping).

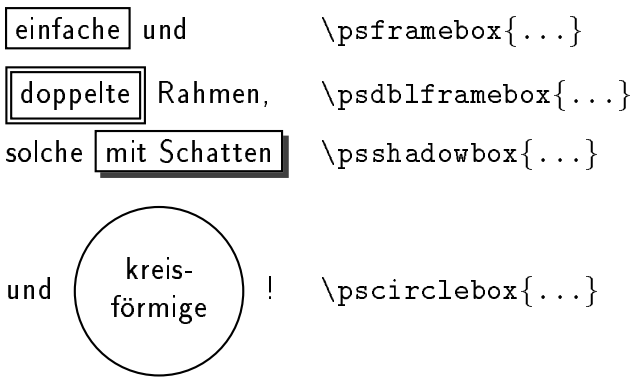
In der neuen Umgebung gibt es einige `put`-Varianten, die in der Regel die Angabe eines Drehwinkels und eines Bezugspunktes zulassen. Probieren Sie mal `\rput` aus:


```
und   \rput{30}(0,0){Auf-}\qquad und
      \rput{-30}(0,0){ab}
```

Es gibt auch `multiput`-Varianten. Schließlich können Sie mit Hilfe der Style-Datei `multido` sogar Zählschleifen (for-next-Typ) mit Variablen verwenden. Ein Beispiel dazu finden Sie in der Anleitung.

4.5 Text-Tricks

Bei den Text Tricks finden wir einfache Befehle, um Texte beliebig einzurahmen. Hier gibt es



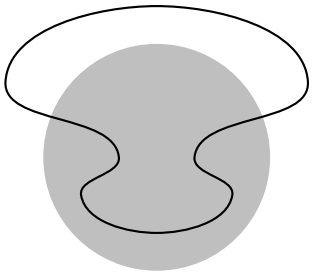
Die gezeigten Rahmen sind TeX in ihrer vollen Größe bekannt, oder anders ausgedrückt, sie sind selbst nochmal in Boxes verpackt. Manchmal kann das stören, dann verwendet man den Parameter `boxsep=false`, wie hier. — Das ging wie folgt:

```
wie \psovalbox[boxsep=false,linecolor=gray]{hier.}
```

Boxes lassen sich nachträglich rotieren und **skalieren**.

```
\rotatedown{rotieren} und \scalebox{2.5 1.5}{skalieren}
```

Hier sehen Sie deutlich, wie schlecht vergrößerte Rasterschriften aussehen! Übrigens können Sie sogar ganze Tabellen oder Bilder rotieren, wie es geht, steht in der Dokumentation von `fancybox.sty`.



Das ist der Umriß:
```
\psccurve(2,0)(4,-1)(2.5,-2)
(3,-2.5)(2,-3)(1,-2.5)(1.5,-2)
(0,-1)
```

Und das die Ausgangsfigur:
```
\pscircle*[linecolor=lightgray]
(2,-2){1.5cm}
```

Und so wird geclippt:
```
\psclip{
\psccurve[linestyle=none]
(2,0)(4,-1)(2.5,-2)(3,-2.5)
(2,-3)(1,-2.5)(1.5,-2)(0,-1)}
\pscircle* ... \endpsclip
```

Abbildung 4.4: *Clipping mit PSTricks*

PostScript besitzt die Fähigkeit zum *Clipping* mit beliebigen Umrissen (das bedeutet, daß nur das Innere des Umrisses sichtbar gelassen wird). PSTricks reicht diese Fähigkeit an Sie weiter. Es gibt die Kommandos `\psclip` und `\endpsclip`. Nach dem ersten stehen einfache Grafikbefehle, die den Umriß definieren, in geschweiften Klammern, dann kommt die Grafik, die maskiert werden soll. Das zweite Kommando schließt die Grafik ab. (Leider scheint es nicht mit Text zu funktionieren. Timothy van Zandt schreibt auch in seiner Anleitung, daß man sich nicht 100%ig auf das Clipping verlassen kann. Sein Beispiel mit Text klappt bei mir nicht.) Ein Beispiel mit Grafik sehen Sie in Abbildung 4.4.

4.6 Diagramme und Knoten

Als letzten Teil der Kurzvorstellung komme ich zu den bemerkenswerten Fähigkeiten von PSTricks beim Umgang mit Diagrammen, die aus einzelnen Elementen und Verbindungen zwischen diesen bestehen. Dafür ist der Zusatz-Style `pst-node` erforderlich. Für Details schauen Sie lieber in die Originalanleitung, hier wieder nur ein „Vorgeschmack“.

Die einzelnen Elemente heißen Knoten (*Nodes*). Sie können eine beliebige TEX-Box nehmen und daraus einen Knoten erzeugen. Dieser Knoten muß optisch nicht hervortreten; er kann aber genausogut in irgendeine Umrahmung (Kreis, Rechteck...) gebracht werden.

Jeder Knoten bekommt bei der Erzeugung einen symbolischen Namen, der nur Buchstaben und Ziffern enthalten darf. Später werden diese Namen benutzt, um die Verbindungen zu zeichnen. Dabei können Knoten aus beliebigen Textteilen (Haupttext, Fußnoten, Tabellen, Abbildungen,

Kopfzeilen) miteinander verbunden werden, solange das zu verbindende Knotenpaar auf ein und derselben Seite liegt. Denn die knotendefinierenden Kommandos merken sich unabhängig vom Kontext bei der Ausführung nur die aktuelle Position des Objekts.

Kommandos zum Erzeugen von Knoten lauten z.B.

```
\rnode[Bezugspunkt]{Name}{Inhalt}
\circlenode{Name}{Inhalt}
\ovalnode{Name}{Inhalt}
\pnode(x,y){Name}
```

Die ersten drei Kommandos verpacken den angegebenen Inhalt in ein Rechteck, einen Kreis bzw. ein Oval. Das Kommando `\pnode` erzeugt dagegen einen Knoten ohne Ausdehnung. Weitere Kommandos enthalten zusätzlich Positionierungsangaben.

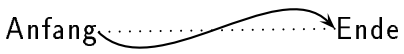
Es gibt mehrere Möglichkeiten, Verbindungen zu ziehen. Das können gerade oder abgewinkelte oder geschwungene Linien sein. Knoten können auch mit sich selbst verbunden werden. Die Linie muß die Knoten aber nicht berühren: Der Parameter `nodesep` regelt den Mindestabstand. Kommandos zum Zeichnen von Verbindungen sind z.B.

```
\ncline{Arrows}{NameA}{NameB}
\nccurve{Arrows}{NameA}{NameB}
\ncarc{Arrows}{NameA}{NameB}
```

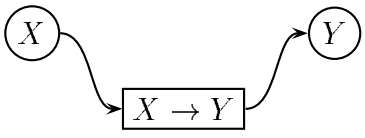
Diese Verbindungen sind gerade Linien, geschwungene Linien (bei denen die Winkel, unter denen die Linie in die beiden Objekte einmündet, durch die Parameter `angleA` und `angleB` bestimmt werden) und sanft gebogene Linien.

Zwei kleine Beispiele:

```
\rnode[r]{A}{Anfang}\hspace{3cm}
\rnode[l]{B}{Ende}
\ncline[linestyle=dotted]{A}{B}
\nccurve[angleA=-45,angleB=135]{->}{A}{B}
```




```
\cnodeput(0,1){ElementX}{$X$}
\cnodeput(4,1){ElementY}{$Y$}
\rput(2,0){\rnode{Kasten}
{\psframebox{$X\to Y$}}}
\nccurve[angleB=180]{->}{ElementX}{Kasten}
\nccurve[angleA=180]{->}{Kasten}{ElementY}
```



Und ein komplexeres Beispiel ohne den Quelltext: Das Diagramm (Abb. 4.5) soll Ihnen einen Eindruck vermitteln, welche Dateitypen (eingekreist) bei der Arbeit mit TEX eine Rolle spielen. Dabei sind die für LATEX typischen Dateien mit gepunktenen Linien versehen. Der grau unterlegte Mechanismus zur Font-Erzeugung wird automatisch gestartet, indem xdvi bzw. dvips bei fehlenden Schriften ein Script namens `MakeTeXPK` aufrufen.

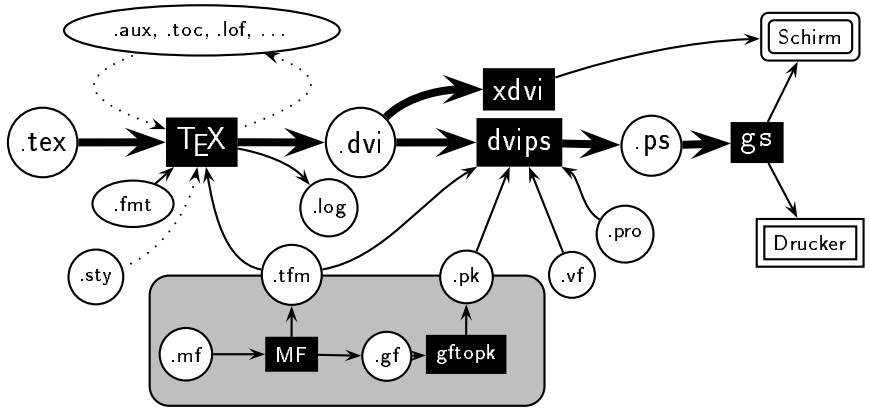


Abbildung 4.5: *Das Ineinandergreifen von Dateien und Programm bei der Arbeit mit TEX*

Mit PSTricks kein Problem!

Kapitel 5

Einbindung von Grafik

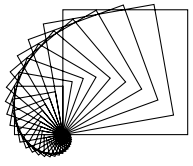
Wenn Sie schon den einen oder anderen Blick in die Literatur zu TEX geworfen haben, wird Ihnen wahrscheinlich schon aufgefallen sein, daß das Thema „Grafik und TEX“ entweder totgeschwiegen oder nur stiefmütterlich behandelt wird (wie bei der `picture`-Umgebung in LATEX). Das liegt daran, daß es in TEX nicht vorgesehen war, Grafiken in den Text einzubauen.

In einer `.dvi`-Datei sind ja nur zwei Arten von Befehlen vorgesehen (wenn man von `\specials` absieht), nämlich Text in der aktuellen Schriftart zu setzen und ausgefüllte Rechtecken mit bestimmten Abmessungen zu zeichnen. Keine schrägen Linien, Kreise oder sonstige Grafikbefehle, keine Farbe oder Graustufen, und erst recht keine Rastergrafiken. Nicht einmal die Schriften werden in einer `.dvi`-Datei genauer als mit ihrem Namen und ihrer Größe beschrieben. Es ist Sache des Treibers, sich für jede Schrift die gerasterten Pixelmuster zu den Zeichen zu besorgen. Solche Rasterschriften sind, wie bereits erwähnt, die `.pk`-Dateien, die bei emTEX meist noch in Fontlibraries versteckt sind.

Man kann `.pk`-Dateien auch als Format für Rastergrafik (s.u.) ansehen. Das ist im Gegensatz zu PostScript eine Möglichkeit, Bilder unabhängig vom `.dvi`-Treiber zu verarbeiten. Es wird eine `.pk`-Datei für jedes Bild erzeugt. Dabei muß ein Bild in Teile zerlegt werden, falls es zu groß für ein Zeichen ist, und hinterher aus mehreren Zeichen zusammengesetzt werden. Die Bilder bleiben damit natürlich auflösungsabhängig. F. Sowa hat für diese Methode das Programm `bm2font` entwickelt, das Sie z.B. auf dem DANTE-Server finden können (`ftp.dante.de`). Doch darum soll es hier nicht gehen.

Vielmehr sind in diesem Text schon einige Vorzüge herausgestellt worden, in deren Genuß Sie durch das Gespann TEX & PostScript gelangen. Auch die Einbindung von Grafiken gehört zu den Möglichkeiten, die sich Ihnen mit PostScript eröffnen. Bevor wir damit anfangen, brauchen Sie allerdings eine Grafik. In den nächsten Abschnitten erfahren Sie manches zum Thema Grafikdateien. Auf die Schnelle können Sie aber auch folgende Zeilen in eine Datei schreiben und diese als „Testbild“ für die Abschnitte 5.3 und 5.4 verwenden:

```
%!PS-Adobe-2.0
%%BoundingBox: 53 82 200 205
```



```
gsave 100 100 translate 20 20 scale newpath
30 {0 0 moveto 5 0 lineto 5 5 lineto 0 5 lineto
closepath 10 rotate 0.9 0.9 scale} repeat
stroke grestore
```

5.1 Einiges zu Grafikdateien

Es gibt mehrere Wege, Bilddateien zu erzeugen. Oft werden im wissenschaftlichen Bereich zwei- und dreidimensionale Graphen benötigt. Besonders für zweidimensionale Darstellungen eignet sich Gnuplot (aktuelle Version 3.5), das sowohl LaTeX-Ausgabe (mit Befehlen der `picture`-Umgebung, mit *TPIC*-Specials, *emTeX*-Specials[1] oder mit PSTricks-Anweisungen) als auch PostScript-Ausgabe unterstützt. Für komplexere wissenschaftliche Abbildungen eignet sich z.B. das kommerzielle Programm Mathematica, das Formeln im TeX-Format und Grafiken in PostScript ausgeben kann, oder Datenvisualierungs-Software wie IDL oder UNIRAS. Schließlich gibt es eine Unzahl an Grafik-/Design-Programmen wie CorelDraw! oder AutoCAD; bei jedem dieser Programme ist PostScript-Ausgabe vorgesehen.

Falls Sie bereits eine Vorlage haben, die Sie als Bild gerne in Ihr Dokument oder Ihre Folien integrieren wollen, ist der einfachste Weg, die Vorlage mit einem Schwarz-Weiß- oder Farbscanner einzuscannen.[2] Die Steuer-Software kann die digitalisierten Bilder in verschiedenen Formaten auf Dateien schreiben.

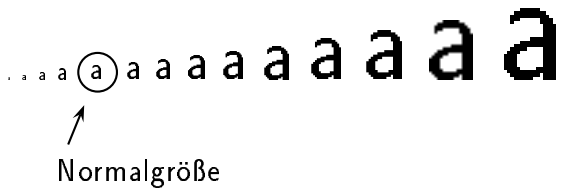
Nun müssen Sie wissen, welches Format für Grafikdateien Sie für ihren Zweck verwenden sollten. Die Auswahl ist groß, und je nach Anwendungsbereich gelten ganz unterschiedliche de-facto-Standards. Darum ist es angebracht, etwas bei dieser Thematik zu verweilen.

Raster- kontra Vektorgrafik

Prinzipiell können Grafiken auf zweierlei Weise beschrieben werden, durch ein Punkteraster ähnlich wie Fotos in Zeitungen, dem eine bestimmte Auflösung zugrunde liegt, oder durch Angabe des Inhalts (Linien, Kreise und „höhere Objekte“). Die erste Methode bezeichnet man mit Rastergrafik, die zweite mit Vektorgrafik. Der Ausdruck kommt daher, daß Vektorgrafiken ursprünglich auf Listen von Vektoren zurückgehen: „Gehe 1cm nach oben, dann 5mm nach rechts und 1mm nach unten, dann...“. Solche Anweisungen sind leicht skalierbar, man muß nur alle Koordinatenangaben mit dem gleichen Faktor strecken oder stauchen. Dagegen kann man Rastergrafiken nicht so schön vergrößern, wie Sie an dem folgenden Beispiel sehen können. Nur die verkleinerten Zeichen sehen noch halbwegs akzeptabel aus:

[1] Das ist der schnellste Weg unter DOS, da der normale emTeX-Previewer das darstellen kann.

[2] und nach Möglichkeit zu vektorisieren, ein langwieriger Prozeß, dessen Ergebnis aber Platz und Zeit bei der Ansicht und beim Druck spart. Dazu eignet sich vor allem das Programm 'CorelTrace', das auch farbige Bilder behandeln kann.



Dennoch sind Bilder meist als Rasterdateien abgespeichert. Das entspricht der Struktur von Bildschirmen und Druckwerken. Schwarzweißbilder werden zeilenweise als Bitmuster abgelegt. Bei farbigen Bildern oder Bildern mit Graustufen legt man den Farbwert eines Punktes in Zahlen des Rot-, Grün- und Blau-Anteils fest (true color), oder man verteilt Codezahlen für jede auftretende Farbe, die wiederum in einer beigefügten Tabelle definiert werden (color maps). Es gibt trotzdem nicht nur ein Format für Rastergrafik, denn dummerweise haben sich verschiedene Leute verschiedene Weisen ausgedacht, z.B.

die Bildgröße oder die Anzahl der Farben des Bildes mitzuteilen. Und dann sind solche Bilder noch sehr groß, eine schwarzweiße DIN-A4-Seite in Laserdrucker-Qualität (300 dpi) braucht ca. 1 MB Speicher.

Kompression

Daher werden die Dateien nach verschiedenen Verfahren komprimiert. Von Bedeutung sind vor allem die folgenden Verfahren:

- RLE (Run-Length-Encoding). Lange Folgen der gleichen Farbe werden abgekürzt: Statt weiß-weiß-weiß-weiß nur 4×weiß.

- Packalgorithmen (Lempel-Ziv & Welch, Huffman-Encoding), wie sie auch `lharc`, `zoo` oder `zip` benutzen. Hier werden für häufig vorkommende Zahlen kürzere Bitmuster benutzt als für seltene Zahlen. RLE und dieses Verfahren sind verlustfrei, d.h. das Ausgangsbild wird identisch rekonstruiert.

- JPEG, ein kombiniertes Frequenzanalyse- und PackVerfahren, das jeweils auf kleinen Teilquadraten der Grafik die Amplituden der schneller und langsamer variierenden Farbanteile bestimmt, oberhalb einer vorgegebenen Frequenzgrenze abschneidet und schließlich die Amplituden mit Packalgorithmen behandelt. Durch das Abschneiden geht zwar Information verloren, aber der Unterschied zum Original ist selbst bei einem Verlust von 25% gering.

- Wavelet-Transformationen, bei denen man Amplituden von selbstähnlichen, orthogonalen Basisfunktionen auf verschiedenen Skalen berechnet und darauf wieder Abschneide- und Packalgorithmen anwendet. Je nach Wahl des Funktionensatzes kann hier mehr Detailtreue als beim JPEG-Verfahren erreicht werden, da auch auf grober Skala hohe Frequenzanteile beigemischt sind.

Einige Grafikformate

Mittlerweile unterscheiden wir Raster- und Vektorformate, bei den Rasterformaten komprimierte und unkomprimierte Formate, bei den komprimierten Formaten verlustfreie und verlustbehaftete Formate. Zusätzlich ist es von Bedeutung, wieviele Farben wiedergegeben werden können. Man spricht von der Farbtiefe in Bits, 1 Bit erlaubt nur die Unterscheidung schwarz/weiß, 8 Bit entsprechen 256 Farbeinträgen in einer Colormap oder 256 Grautönen, 24 Bit gestatten Echtfarbdarstellung (16 Mio. Farben).

Um nun herauszufinden, welche Formate aus der Vielfalt von `PCX`, `BMP`, `IFF/ILBM`, `TGA`, `GIF`, `PIC`, `TIFF`, `RGB`, `GEM`, `XWD`, `JPG`, `IMG`, `PPM` usw. eigentlich geeignet und verbreitet sind, kann man auf FTP-Servern herumschnüffeln oder in die Import- und Export-Funktionen gängiger Programme schauen. Sie sollten sich als Fazit folgendes merken:

- `GIF` ist bis zu einer Anzahl von 256 Farben das Rasterformat ihrer Wahl, wenn es um gute, verlustfreie Datenkompression geht. Das heißt, aus einer `.gif`-Datei läßt sich das Original 100%ig rekonstruieren.

- `JPG` lautet die Alternative, wenn Sie kleine Verluste in der Detailtreue hinnehmen können. Die Verluste sind bei Fotos kaum merklich, bei Vorlagen mit scharfen Konturen dagegen indiskutabel.

Tatsächlich finden Sie auf FTP-Servern fast nur `.gif`- oder `.jpg`-codierte Bilder. Dagegen werden Sie beim Verarbeiten eigener Vorlagen feststellen, daß die Scanner-Software ausgerechnet diese Formate nicht unterstützt, dafür aber `BMP`, `TIFF` oder andere anbietet. Dazu gleich ein Wort der Warnung:

- Wenn Sie das `TIFF`-Format benutzen müssen (z.B. weil `GIF` nicht unterstützt wird, oder weil `TIFF` immerhin die volle Farbanzahl von 16 Mio. Farben gestattet), **nehmen Sie nie eine komprimierende TIFF-Variante!** Davon gibt es mehrere, so daß Sie Ihre Grafikdatei woanders womöglich nicht verarbeiten können.

Gegen unkomprimierte `.tif`-Dateien ist nichts einzuwenden außer ihre unhandliche Größe. Sie müssen zum Transport auf Diskette mit Packern wie `lharc` komprimiert werden (ein buntes Bild von 10cm$\times$6cm in voller Farbzahl und 300 dpi Auflösung würde sonst schon 2 MB belegen).

Keines der genannten Formate ist geeignet, direkt von dvips in eine PostScript-Datei eingesetzt zu werden. Sie haben aber im Kapitel über PostScript bereits EPS-Dateien kennengelernt. Solche Dateien können von dvips direkt in die PostScript-Ausgabedatei integriert werden.

5.2 Software zur Grafikbearbeitung

Kommen wir zurück zu der inzwischen eingescannten Vorlage oder der Grafikdatei aus einem Zeichen- oder Datenvisualisierungsprogramm. Hier sind viele Wünsche offen: Sie wollen sich das Bild noch mal auf dem Bildschirm betrachten, die Ränder müssen gestutzt werden, Sie wollen es drehen oder verzerren, ein farbiges Bild für den Schwarzweißdruck aufbereiten, oder Sie brauchen es schlicht in einem anderen Format. Wobei das schließlich das EPS-Format sein wird.

Es gibt glücklicherweise genügend Programme, die all diese Funktionen bieten. Die zwei folgenden Abschnitte nennen Software unter MS-DOS und MS-Windows bzw. unter Unix. Diese Auswahl erhebt nicht den Anspruch auf Vollständigkeit.

Wenn Sie andere Programme benutzen, ist der Schwachpunkt oft die für uns so wichtige PostScript-Ausgabe. Das **pbmplus**-Paket erzeugt beispielsweise PostScript-Dateien für ganzseitige Ausgabe des Bildes auf einem entsprechenden Drucker. Fast immer geht es nur darum, das Bild möglichst formatfüllend auf der Seite unterzubringen, und Angaben wie die Bounding Box werden weggelassen.

Software für MS-DOS und MS-Windows

Bekannte Grafikprogramme sind **Graphic Workshop** und **Alchemy**. Beide Programme sind Shareware, d.h. Sie können sich die Programme zum Ausprobieren umsonst kopieren, müssen aber eine Lizenzgebühr zahlen und sich registrieren lassen, wenn Sie beabsichtigen, das jeweilige Programm ernsthaft zu nutzen.

Graphic Workshop gibt es in der Version 6.1v für MS-DOS und wird über Menüs recht komfortabel bedient. Es gibt auch eine Windows-Version, die im Gegensatz zur MS-DOS-Version keine EPS-Dateien erzeugen kann. Das läßt sich normalerweise verschmerzen, macht sie aber für unsere Zwecke ungeeignet.

Alchemy ist für MS-DOS gedacht und wird über die Kommandozeile mit verschiedenen Parametern aufgerufen. Die unregistrierte Version kann nur Bilder bis zu einer Größe von 640×480 Punkten bearbeiten.

Es gibt noch eine Möglichkeit, unter MS-Windows EPS-Dateien zu erzeugen. Sie nehmen einen passenden PostScript-Druckertreiber, wie er etwa bei einem HP Laserjet 4ML mitgeliefert wird, und stellen ihn auf Dateiausgabe und EPS-Format ein. Dann können Sie aus beliebigen Programmen drucken, wobei mir allerdings nicht bekannt ist, was mit farbigen Vorlagen geschieht.

Software für Unix

Ein bekanntes Programm für das X-Window-System ist **xv** von John Bradley (Version 3.0). Es ist vor allem zum Betrachten von Grafiken gedacht. Sie können die Farbpalette auf viele

Arten beeinflussen, z.B. um den Kontrast zu erhöhen oder einen Farbstich zu korrigieren. Da es die wichtigsten Formate der Unix-Welt und TIFF, GIF sowie JPEG unterstützt, können Sie xv gut zum Konvertieren einsetzen. Schließlich ist außer der komfortablen PostScript-Ausgabe die PostScript-*Eingabe* zu erwähnen, xv ruft in solchen Fällen GhostScript im Hintergrund auf. xv ist ebenfalls Shareware.

Bis auf die grafische Darstellung am Bildschirm ist man auch mit dem **pbmplus**-Paket gut bedient. Diese Software verwendet Zwischenformate und stellt Konvertierungsprogramme (als Filter) von allen gängigen Formaten in die Zwischenformate und von dort aus weiter in wiederum alle gängigen Formate zur Verfügung. Auf dem Zwischenformate können diverse Operationen vorgenommen werden, z.B. Abschneiden, skalieren, andere Transformationen, Gamma-Korrektur, Verbinden mehrerer Bilder, Effekte wie Strukturen betonen, Ölgemälde, Relief, Textur darüberlegen, Weichzeichner, Farbquantisierung. Nachteilig bleibt die unbequeme Art der Bedienung, die ein gutes Gedächtnis für die vielen Programmnamen voraussetzt.

Schließlich ist das **ImageMagick**-Paket wie die beiden anderen auch als C-Quelltext erhältlich. Es bietet sowohl die Kommandozeileneingabe (mit `convert` und `mogrify`), die für Shell-Skripte unerläßlich ist, als auch eine ansprechende X11-Oberfläche (die Programme `display` und `animate`). Im Vergleich zu xv liegt der Schwerpunkt mehr auf der Bildbearbeitung. ImageMagick und pbmplus sind keine Shareware, sondern Public Domain.

Bemerkungen zur Konversion ins EPS-Format

Sie haben die Grafik mit dem Programm Ihrer Wahl geladen, wählen das Zielformat EPS oder PostScript und lassen konvertieren... Da in PostScript beliebige Farben erlaubt sind, könnte Ihre Grafik im Prinzip 1:1 umgesetzt werden. Doch das scheitert entweder am Konvertierungsprogramm, das nur Schwarzweiß-PostScript schreiben mag, oder am Schwarzweiß-Ausgabegerät, das Farben nur gerastert wiedergibt. In der Regel kommt eine Helligkeitsschwelle, die alle dunkleren Bildpartien schwarz und alle helleren weiß färbt, nicht in Frage, denn damit geht außer bei Strichzeichnungen jeglicher Detailgehalt der Bilder verloren.

Daher wird man üblicherweise einen Prozeß namens „Dithering“ anwenden, um die Graustufen oder Farben des Ausgangsbildes in Rastermuster aus schwarzen und weißen Punkten umzusetzen. Da gibt es z.B. geordnete Muster, die sich sehr schnell erzeugen lassen. Das können 4×4-Matrizen sein, deren sechzehn Felder mit 0 bis 16 schwarzen Punkten ausgefüllt sind, der Rest ist weiß. Aus jedem Bildpunkt wird so eine Matrix, man kann 16 Graustufen darstellen. Aber es sieht eben doch zu regelmäßig aus.

Interessanter sind Diffusionsverfahren, die aus dem Grauwert eine Wahrscheinlichkeit bilden, den schwarzen Punkt oder den weißen zu setzen. Solche Rasterungen enthalten das regelmäßige Element nicht mehr, auf das unser Sehapparat so empfindlich reagiert. Immer zu empfehlen ist Floyd-Steinberg-Dithering.

Ein konkretes Problem und dessen Lösung

Abschließend will ich Ihnen noch eine Anregung geben, wie Sie die verschiedenen Grafikprogramme im Zusammenspiel einsetzen können. Gegenstand der Diskussion ist die Abbildung 1.3 auf Seite 6, die mir anfangs große Schwierigkeiten bereitet hat.

Es war natürlich naheliegend, innerhalb einer `tabular`-Umgebung die Schriften nacheinander aufzurufen und zweispaltig untereinander zu setzen. Dieses Verfahren funktionierte auf meinem Linux-System zuhause sofort, auch ein Ausdruck war auf dem BubbleJet mit GhostScript kein Problem. Jedoch fand ich keinen PostScript-Laserdrucker, der mit dieser Seite fertig geworden wäre. Schlimmer noch, das ganze Dokument war nicht mehr druckbar!

Offenbar war die große Anzahl von PostScript-Schriften schuld. Kein Drucker konnte 47 Schriften gleichzeitig verwenden, aber dvips veranlaßt gleich am Anfang der PostScript-Datei das Laden aller Schriften. Daher produzierten die Drucker nicht einmal die erste Seite. Da ich nicht auf die Schriftenliste verzichten wollte und außerhalb der Tabelle nur wenige PostScript-Schriften auftauchen, beschloß ich, die Tabelle als Rastergrafik einzubinden.

Als erstes legte ich die Zielauflösung fest: 300 dpi. Dann wurde die Tabelle ohne Rahmen in eine separate TEX-Datei geschrieben. Nun enthielt die Seite nur noch die nackten Schriftproben. Mit GhostScript wandelte ich die PostScript-Datei in eine Rasterdatei um. Um deren Größe klein zu halten, hatte ich die Tabelle in die linke untere Seitenecke gebracht und konnte das Papierformat DIN A5 verwenden:

```
gs -r300 -sDEVICE=pbmraw -sPAPERTYPE=a5
   -sOutputFile=stdfonts.pbm stdfonts.ps
```

Die entstandene Rastergrafik war etwa 1700×2500 Pixel groß. Jetzt sollten die weißen Ränder soweit wie möglich abgeschnitten werden, um das Rechteck, daß danach ins EPS-Format konvertiert werden sollte, so klein wie möglich zu machen. Zuerst versuchte ich, die Grafik mit xv zu laden und dessen Funktion *AutoCrop* zum Abschneiden zu verwenden. Mit nur 8 MB RAM dauerte das Laden der Datei schon etliche Minuten, das Abschneiden brach ich nach einer Stunde ununterbrochenen Festplatten-Pagings ab.

Die Schwierigkeit ist, daß xv nur einen 8-bit- und einen 24-bit-Modus kennt. Mit 8 Bits pro Pixel benötigt die Rasterdatei etwa 4.3 MB. Ich brauchte also ein Programm, das pure Schwarzweiß-Grafik bearbeiten kann: `pnmcrop` (aus dem pbmplus-Paket). Mit

```
pnmcrop stdfonts.pbm >stdfontc.pbm
```

konnte ich zusehen, wie die Grafik zuerst durchgescannt wurde, bald die Größe der abgeschnittenen Streifen ausgegeben wurde und schließlich die Ausgabedatei entstand. In etwa zwei Minuten lag eine noch etwa halb so große Datei vor, die ich nun mit xv laden und nach PostScript konvertieren konnte. Die entstandene EPS-Datei war 490 KB groß; die Mühe hatte sich gelohnt.

Als Fazit bleibt festzuhalten, daß gerade im Bereich Grafikbearbeitung ein einzelnes Tool oft nicht ausreicht. Obwohl xv durch seine leichte Bedienung brilliert und für mich stets das „Programm der Wahl“ ist, konnte es hier nicht eingesetzt werden. Und umgekehrt zeigte das

pbmplus-Paket, daß Bedienungskomfort nicht immer an erster Stelle steht, sondern Sparsamkeit im Umgang mit Ressourcen in diesem Fall wesentlich zum Erfolg beiträgt.

5.3 Der Style epsf

Der Autor von dvips, Tomas Rokicki, hat dem Programm einen Style namens `epsf` beigelegt.[3] Mit diesem Style können Sie im Text durch das Kommando

`\epsfbox{`*EPS-Dateiname*`}`

beliebige EPS-Dateien einbinden (als vbox). In der `center`-Umgebung müssen Sie das Kommando `\leavevmode` vorausschicken. Mit dem Kommando `\epsfverbosetrue` können Sie verfolgen, in welcher Größe die Abbildungen eingebunden werden. Falls die EPS-Datei keine `BoundingBox`-Eintragung besitzen sollte, ist das Bild nicht skalierbar, sondern wird in seiner „natürlichen“ Größe eingesetzt. Falls Sie zuvor

```
\epsfverbosetrue
```

gesetzt haben, meldet `epsf` in diesem Fall

```
! No bounding box comment in ...; using defaults
```

Sie können die `BoundingBox` aber mit `bb.ps` ermitteln (siehe Abschnitt 2.6) und dann in eckigen Klammern angeben:

`\epsfbox[`*llx lly urx ury*`]{...}`

Die Abkürzungen bedeuten „lower-left-x/y“ und „upper-right-x/y“ und bezeichnen die zwei gegenüberliegenden Eckpunkte des `BoundingBox`-Rechteckes in $1/72$-Zoll-Einheiten (PostScript-Koordinaten). Eigene Angaben haben hier Vorrang, in der Datei wird dann nicht mehr nachgesehen.

Sie können die Abbildung mit Kommandos skalieren, die Sie am besten direkt vor dem Kommando `\epsfbox` geben. `\epsfxsize` und `epsfysize` legen jeweils nur die Breite bzw. nur die Höhe fest, das Bild wird dabei unverzerrt verkleinert. Die größte Flexibilität bietet `\epsfsize`, welches Sie so umdefinieren müssen, daß es die zu verwendende Größe in x-Richtung zurückgibt. Die beiden Argumente sind die natürliche Größe des Bildes in x- und y-Richtung. Besonders interessant ist z.B. die Definition

```
\def\epsfsize#1#2{\ifnum#1>\hsize\hsize\else#1\fi} ,
```

mit der die x-Breite des Bildes nicht größer als die Textbreite werden kann.

Die Abbildung kann außerhalb der `BoundingBox` abgeschnitten („geclippt“) werden, indem Sie `\epsfclipon` angeben. Das Gegenstück zum Abschalten heißt `\epsfclipoff`. Für Entwurfszwecke können Sie das langwierige und speicherintensive Einbinden der Grafiken mit

[3] `epsf` läßt sich auch in plainTEX nutzen.

`\epsfdrafttrue` und `\epsfdraftfalse`

abschalten bzw. wieder einschalten.

Wenn `epsf` die angegebene Datei nicht finden kann, meldet es den Fehler

```
! I couldn't open ..., will ignore it.
```

Sollte Ihre Version des Styles `epsf` nur das Kommando `\epsffile`, aber nicht `\epsfbox` verstehen, handelt es sich um eine veraltete Version, die Sie ersetzen sollten.

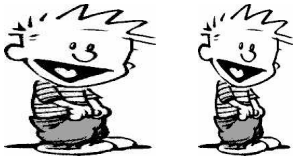
5.4 Der Style epsfig

Dieser Style ist entstanden aus einer Kombination von den Styles `epsf` und `psfig`. Dabei wurden die Funktionen von `epsf` genommen, wobei das Kommando `\leavevmode` automatisch bei Bedarf gegeben wird, die Bedienung stammt von `psfig`. Die Fehlerbehandlung wurde verbessert. Hier müssen Sie sich nicht so viele einzelne Kommandos merken. Das wesentliche ist in dem folgenden Befehl zusammengefaßt:

```
\epsfig{file=...,
   height=..., width=...,
   bbllx=..., bblly=..., bburx=..., bbury=...,
   rheight=..., rwidth=...,
   angle=..., clip=, silent=}
```

Sie sehen in der ersten Zeile die Angabe der Datei, und die ist natürlich Pflicht! Alle anderen Angaben sind optional.

Mit den nächsten Parametern können Sie Breite und/oder Höhe des Bildes vorgeben. Wenn Sie beides angeben, hält `epsfig` sich daran; wenn Sie nur Breite oder Höhe angeben, wird das Bild so skaliert, daß die Proportionen erhalten bleiben. Vergleichen Sie die Abbildungen...



```
\epsfig{file=calvinsw.eps,
   width=2cm, height=2cm}
\epsfig{file=calvinsw.eps,
   height=2cm}
```

Anschließend finden Sie die Parameter, um eine fehlende Bounding Box anzugeben, die Sie nachträglich mit mit `bb.ps` ermittelt haben.

Unter Unix gibt es eine zweite Möglichkeit. Wenn `epsfig` die angegebene Datei nicht finden kann, hängt es die Endung `.bb` an den Namen an und sucht die entsprechende Datei in der Hoffnung, dort einen `BoundingBox`-Kommentar zu finden. Bei Erfolg gibt es eine Meldung aus:

```
using BB from Bilddatei.bb
<"'epsfig Bilddatei">
```

Abbildung 5.1: *Der Autor, rotiert mit epsfig. Der Platzbedarf (die umschriebene Box) wird normalerweise nach der Rotation ermittelt.*

In den spitzen Klammern steht der `\special`-Befehl, der an dvips gegeben wird, d.h. dvips soll die Ausgabe des Kommandos `epsfig` in die PostScript-Datei einbauen (siehe Abschnitt 3.4, Seite 29). Dieses Kommando könnte im einfachsten Fall aus einem Shell-Script mit dem Inhalt

```
#!/bin/sh
zcat $1.eps.gz
```

bestehen, womit ein komprimiert vorliegendes Bild leicht eingebunden werden könnte.

Bei `epsfig` sind zwei Perl-Scripts dabei, eins namens `epsbb`, welches aus einer PostScript-Datei die `BoundingBox` herausliest und in eine Datei mit der angehängten Endung `.bb` schreibt, und ein anderes namens `epsfig`, das nach einer Datei mit der Endung `.gz`, `.z` oder `.Z` sucht (und zwar in den Verzeichnissen, die in der Variable `TEXINPUTS` aufgeführt sind) und diese dann entkomprimiert ausgibt. Die Krönung ist das TEX-Kommando `\pscompress`, mit dem beim allerersten dvips-Durchlauf einer TEX-Datei alle PostScript-Bilder automatisch mit `epsbb` behandelt und dann komprimiert werden.

Normalerweise reserviert `epsfig` genau soviel Platz, wie das Bild (bzw. dessen `BoundingBox`) braucht. Wenn Sie mehr oder weniger Platz reservieren wollen, verwenden Sie die Angaben `rheight` und `rwidth`.

Zum Rotieren des Bildes dient `angle`. Dabei wird soviel Platz reserviert, wie das kleinste aufrecht stehende Rechteck braucht, das das gedrehte Bild enthält (siehe Abb. 5.1). Wenn Sie gleichzeitig mit `width` oder `height` eine Größe vorgegeben haben, bezieht sich diese entweder auf die Größe

vor oder nach der Rotation (Voreinstellung). Dieses Verhalten können Sie mit

```
\psscalefirst und \psrotatefirst
```

umschalten.

Schließlich können Sie das Bild mit der BoundingBox clippen, also alles abschneiden, was herausragt. Dazu müssen Sie nur 'clip=' angeben.

Auch bei epsfig können Sie in der Entwurfsphase eines Dokuments mit den zusätzlichen Kommandos

```
\psfull und \psdraft
```

nach Belieben im Dokument umschalten, ob EPS-Dateien voll eingebunden werden oder ob ersatzweise nur ein Rahmen von der Größe der BoundingBox dargestellt wird.

Zu den Fehlermeldungen: Wenn die angegebene Bilddatei nicht existiert, erscheint die Meldung

```
! ERROR. PostScript file ... not found.
```

Kann epsfig keine BoundingBox finden, meldet es

```
! ERROR - Cannot locate BoundingBox.
```

5.5 Wie Text eine Grafik umfließen kann

Kann man Text um Grafik herum fließen lassen? Da gibt es leider keine absolut zuverlässige Methode, aber Sie können ja mal den Style floatfig ausprobieren. Damit sehen die Befehle für ein umflossenes Bild am Rand so aus:

\begin{floatingfigure}{*Bildbreite*}
...Bildmaterial...
\caption{*Bildunterschrift*}
\end{floatingfigure}

Das Argument der floatingfigure-Umgebung ist die Breite des freizuhaltenden Randbereichs. Die Umgebung muß im *vertical mode*, d.h. nicht innerhalb eines Absatzes stehen. Das Bild landet links oder rechts, je nachdem, ob die aktuelle Seite eine gerade oder ungerade ist. Die Numerierung verträgt sich mit der figure-Umgebung. Zu Beginn des Dokuments müssen Sie direkt hinter der Zeile \begin{document} den Befehl

```
\initfloatingfigs
```

geben. Meine Erfahrung ist jedoch, daß bei mehr als einem Bild pro Seite fast nur noch Warnungen über kollidierende Abbildungen ausgegeben werden und letztendlich keine Grafiken auf der Seite erscheinen; wenn doch welche auftauchen, ist das ein recht wackeliges Ergebnis, das durch Layoutänderungen schnell wieder verlorengehen kann.

Kapitel 6

NFSS

NFSS, das *New Font Selection Scheme*, ist ein neuer Mechanismus der Fontauswahl für TEX. Es kann in den üblichen Makropaketen wie LATEX, plain TEX, $\mathcal{AMS}$-TEX usw. eingesetzt werden. In der neuesten LATEX-Version, LATEX 2_ε, ist es bereits fester Bestandteil. Bevor ich auf NFSS eingehe (und zwar auf Version 2), will ich einen Überblick über das alte System von TEX-Schriften geben.

6.1 Allgemeines zu Schriften

Es gibt sehr viele verschiedene Schriften für die unterschiedlichsten Zwecke. Plakat- und Zierschriften sind auf Auffälligkeit hin ausgerichtet, sehr individuell und daher besonders schwer zu klassifizieren. Textschriften sollen dagegen auf möglichst angenehme, unauffällige Weise große Textmengen darstellen. Dabei werden immer drei Varianten benötigt, eine Variante für den normalen Text, eine für Hervorhebungen und eine für Überschriften. Hervorhebungen werden durch *geneigte* (*oblique*) oder durch *kursive* (*italic*) Schrift gekennzeichnet. Überschriften werden meist *fett* oder *breit* gesetzt. Als weiteres Merkmal tritt die Schriftgröße hinzu, die sich danach richtet, ob es sich um normalen Text, Überschriften verschiedener logischer Ebenen oder um Fußnoten handelt. Alle diese Schriftvarianten sind Abkömmlinge einer *Schriftfamilie*, die sich durch variantenübergreifende Charakteristika auszeichnet.

Jede Variante wird gewöhnlich eigens entworfen. Das ist besonders deutlich am Unterschied zwischen der normalen und der kursiven Form einer Schrift zu erkennen, aber auch eine fette, schwere Schrift entsteht nicht einfach dadurch, daß alle Linien einer leichteren Variante pauschal dicker gezogen werden; und verschieden große Schriften haben verschiedenen Lesbarkeitsansprüchen zu genügen, so daß simples Vergrößern und Verkleinern nicht ausreicht.

Um Textschriften systematisch zu erfassen, unterscheidet man daher die Attribute

Familie, **Gewicht**, **Breite**, **Form** und **Designgröße**.

Auch Schriftfamilien lassen sich ganz grob einteilen in Schriften mit Serifen,[1] serifenlose Schriften, gebrochene Schriften wie Fraktur oder Schwabacher und Sonderschriften wie z.B. Schreibmaschinenschrift, bei der alle Buchstaben den gleichen Platzbedarf haben.

Serifenlose Schriften wirken geradliniger und moderner als Serifenschriften, obwohl es sie auch schon seit zwei Jahrhunderten gibt; sie besitzen in der Regel keine kursive Variante, sondern eine geneigte Variante, die vollkommen ausreicht, um innerhalb des aufrechten, einheitlichen Schriftbilds Hervorhebungen zu betonen. Dagegen verwenden Serifenschriften durchweg kursive Varianten, weil unter der ohnehin sehr vielseitigen aufrechten Form, meist „roman“ genannt, eine Neigung alleine kaum auffallen würde. Die kursive Variante enthält noch mehr runde, filigrane Elemente und zum Teil andere Buchstabenformen (etwa bei 'a' und 'g') und erreicht so ein hohes Maß an Eigenständigkeit und Auffälligkeit.

6.2 Computerschriften und Codierungen

Computer ordnen den einzelnen Zeichen intern Codenummern zu. Eine Computerschrift enthält daher immer eine Codierung, über die der Symbolvorrat der Schrift angesprochen werden muß. Dabei gibt es verschiedene Standards, die sich im Laufe der Zeit gebildet haben.

- der **ASCII-Code**. Diese Codierung ordnet den Zahlen 32–126 Groß- und Kleinbuchstaben, Ziffern und wichtige Symbole zu. Akzentuierte Zeichen, etwa Umlaute, sind nicht enthalten. Der ASCII-Code ist die Grundlage für etliche internationale Erweiterungen. Allen Erweiterungen ist gemein, daß sie nicht mehr mit sieben Bit (128 Nummern) auskommen.

- **ISO 8859-1 (Latin-1)**. Diese Codierung enthält in den Zeichenpositionen 192–255 die wichtigsten westeuropäischen Sonderzeichen, z.B. alle französischen Akzentkombinationen und die deutschen Umlaute (siehe Tabelle B.4). Die meisten Schriften des X-Window-Systems liegen in dieser Codierung vor. MS-Windows benutzt diese Codierung. Die neue, internationale TEX-Codierung basiert auch auf dieser Codierung (s.u.).

- **IBM Codepages**. Das sind verschiedene Codierungen unter DOS, die im Bereich 128–255 Grafiksymbole und eine Auswahl nationaler Sonderzeichen definieren. Im deutschen Bereich sind die Codepages 437 und 850 von Bedeutung.

- **Unicode**. Unicode ist ein Versuch, lateinische Schriftzeichen mit anderen (asiatischen, arabischen usw.) Schriftzeichen gemeinsam in einem 16 Bit-Code (65536 Zeichen) unterzubringen. Bislang ist die westliche EDV-Welt jedoch noch in das DOS- und das Latin-1-Lager gespalten.

[1] Serifen nennt man die Verzierungen an den Strichenden von Zeichen, die vorwiegend horizontal verlaufen und so die Augen beim Lesen leichter die Zeile entlang führen sollen.

6.3 Schriften in TEX

Zu TEX gehörten von Anfang an ein Satz von Schriften aus der Familie Computer Modern. Diese Schriften wurden von Knuth mit seinem METAFONT-System entworfen. METAFONT arbeitet mit einer Schriftbeschreibungssprache. Ähnlich wie PostScript ein PostScript-Programm mit Grafikelementen, Koordinaten und Arithmetik in eine Rasterdarstellung einer Druckseite umwandelt, erzeugt METAFONT eine Rasterdarstellung einer Schrift.

Die Familie Computer Modern ist in drei Gattungen unterteilt: Roman, Sans Serif und Typewriter (wenn man von den selten verwendeten Schriften wie *Fibonacci* oder *Funny Roman* absieht). In jeder Gattung gibt es einige wenige Varianten. Bei dieser überschaubaren Anzahl wird jede Schrift in TEX durch ein zugeordnetes Kommando angesprochen, z.B. `\tenrm` für die Schrift `cmr10` (Computer Modern Roman in der Designgröße 10 Punkt) oder `\sevenbf` für `cmbx7` (CM Bold Extended in 7 Punkt).

Schriftgröße und Designgröße

Die übliche Designgröße der CM-Schriften ist 10 pt.[2] Für Schriften, die oft auch größer benötigt werden, gibt es eine gesonderte 12 pt- oder gar eine 17 pt-Variante, für kleinere Schriften 9 pt, 8 pt, bis hinunter zu 5 pt.

Außer der Wahl der Schriftgröße durch die passende Designgröße kann in plainTEX das gesamte Dokument mit der Anweisung `\magnification` um einen beliebigen Faktor skaliert werden.

Nehmen wir einmal an, das Ausgabegerät, z.B. ein Laserdrucker, habe eine Auflösung von 300 dpi. Damit der `.dvi`-Treiber die Schrift `cmr10` passend zum Drucker verwenden kann, muß METAFONT eine gerasterte Version in der Auflösung 300 dpi erzeugt haben. Für ein um 20% vergrößertes Druckbild wird eine Rasterschrift mit 360 dpi gebraucht. Da jede Vergrößerungsstufe eigene Rasterschriften benötigt, ist es unsinnig, beliebige Größen zu verwenden. Man einigt sich stattdessen auf eine abgestufte Skala, die von Schritt zu Schritt 20% vergrößert. Die Reihe der Skalierungsfaktoren lautet

1.0, 1.2, 1.44, 1.728, 2.0736, 2.48832, ...

In TEX heißen die einzelnen Schritte `\magsteps`. Für nur geringfügig vergrößerte Texte wird zusätzlich ein halber Schritt eingeführt, der entsprechende Faktor ist 1.0954...

Diese Vergrößerung hat nichts mit der Designgröße zu tun, die bleibt unverändert. Vergleichen Sie eine 10 pt-Schrift mit einer auf doppelte Größe gebrachten 5 pt-Schrift:

Schrift cmr10 Schrift cmr5

[2] Ausgenommen die Schrift *Computer Modern Fibonacci*, die es nur in der Designgröße 8 pt gibt.

LaTeX-Schriftgrößen

LaTeX kennt im Gegensatz zu plainTeX Kommandos, um die Größe der Schrift zu variieren, ohne Gebrauch von der `\magnification` zu machen.

Dabei wird über eine Tabelle versucht, eine möglichst gut passende Designgröße auszuwählen; ein Verfahren, das ohne Zweifel besser als einfache Skalierung ist.

Die Umschaltkommandos zum Vergrößern lauten

`\large`, `\Large`, `\LARGE`, `\huge`, `\Huge`

und entsprechen in ihrer Wirkung der `\magstep`-Reihe. Da es die Schrift `cmr` in den Designgrößen 10, 12 und 17 pt gibt, die Schrift `cmbx` aber nur in 10 und 12 pt, trifft die Schriftauswahltabelle in LaTeX folgende Zuordnung von Designgrößen und Skalierungsfaktoren:[3]

		`\large`	`\Large`	`\LARGE`	`\huge`	`\Huge`
1.0	1.095	1.2	1.44	1.728	2.074	2.488
cmr10 ·1.0	cmr10 ·1.095	cmr12 ·1.0	cmr12 ·1.2	cmr17 ·1.0	cmr17 ·1.2	cmr17 ·1.44
cmbx10 ·1.0	cmbx10 ·1.095	cmbx12 ·1.0	cmbx12 ·1.2	cmbx12 ·1.44	cmbx12 ·1.728	cmbx12 ·2.074

Für den oben genannten Laserdrucker werden also die Rasterschriften

`cmr10` in 300 und 329 dpi,
`cmr12` in 300 und 360 dpi,
`cmr17` in 300, 360 und 432 dpi sowie
`cmbx10` in 300 und 329 dpi,
`cmbx12` in 300, 360, 432, 518 und 622 dpi

benötigt. Da LaTeX aber auch eine Gesamtvergrößerung mit den Stiloptionen `11pt` (eigentlich 10.95...) und `12pt` bei `article`-, `book`- und `report`-Dokumenten vorsieht, erweitert sich die Liste der Rasterdateien noch einmal.

Die Umschaltkommandos zum Verkleinern lauten

`\small`, `\footnotesize`, `\scriptsize` und `\tiny`.

Hier gilt nicht die `\magstep`-Reihe, sondern die Zuordnung

`\tiny`		`\scriptsize`	`\footnotesize`	`\small`	
0.5	0.6	0.7	0.8	0.9	1.0
cmr5 ·1.0	cmr6 ·1.0	cmr7 ·1.0	cmr8 ·1.0	cmr9 ·1.0	cmr10 ·1.0
cmti7 ·5/7	cmti7 ·6/7	cmti7 ·1.0	cmti8 ·1.0	cmti9 ·1.0	cmti10 ·1.0

[3] ohne die Stiloption 11- oder 12 pt

Die Skalierungsfaktoren sind also an ganzzahlige Designgrößen angepaßt. Während die Schrift `cmr` in den Designgrößen 9, 8, 7, 6 und 5 pt vorliegt, gibt es die Schrift `cmti` (CM Text Italics) nur in 10, 9, 8 und 7 pt, so daß die kleineren Schriften durch Skalieren der 7 pt-Schrift mit den Faktoren 5/7 bzw. 6/7 gebildet werden müssen. Das führt zu den Rasterdateien

`cmr5`, `cmr6`, `cmr7`, `cmr8` und `cmr9` in 300 dpi sowie
`cmti7` in 214, 257 und 300 dpi und
`cmti8` und `cmti9` in 300 dpi.

(Die Designgröße 6 pt wird von den Stiloptionen `11pt` und `12pt` verwendet.)

Motivation für NFSS

Innerhalb der gerade aktiven Schriftgröße stellt LaTeX die bekannten Auswahlkommandos für die Schriftvariante zur Verfügung, als da sind

`\rm`, `\bf`, `\it`, `\sc`, `\tt`, `\sf`, `\em`.

Wenn eine neue Schriftgröße gewählt wird, läuft eine Initialisierungssequenz ab, die diese Kommandos auf die neuen Schriften umdefiniert. Die vorher geltende Variante ist vergessen, es wird immer auf `\rm` umgeschaltet. Das führt zu Überraschungen bei LaTeX-Anfängern, denn es ist zunächst völlig unklar, warum es `\Large\bf` und nicht `\bf\Large` heißen muß.

Während TeX mit sehr wenigen Fonts das Licht der Welt erblickte und es daher ausreichte, für jeden dieser Fonts ein eigenes zugeordnetes Kommando zu schaffen, sind inzwischen eine Unzahl zusätzlicher Schriften erhältlich, die in den üblichen Versionen von TeX und LaTeX nur schlecht genutzt werden können. Die starren Tabellen müßten dazu durch frei ladbare Tabellen ersetzt werden, und die Auswahl der zur Anforderung passenden Schrift sollte möglichst berechnet und nicht fest vorgegeben werden, um flexibel zu bleiben. Aus diesen Forderungen entstand das neue Schriftauswahlverfahren NFSS. Die Gelegenheit wurde genutzt, um das ganze Fontkonzept neu zu ordnen und zu systematisieren, z.B. Schriften für den Textsatz von solchen für Formelsatz klar zu trennen.

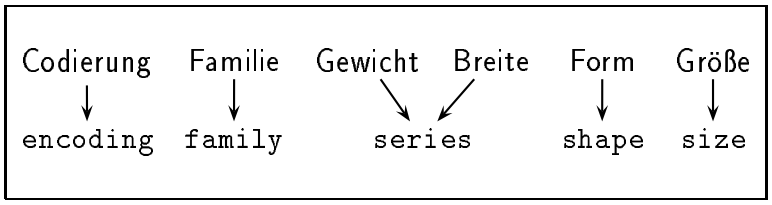
NFSS wurde erstmals 1989 vorgestellt. In diesem ersten Release konnte man bereits bequem die einzelnen Fontattribute unabhängig voneinander auswählen. Allerdings war dort noch nicht berücksichtigt, daß mit TeX 3.0 und den Forderungen nach mehr Internationalität auch verschiedene andere Codierungschemata der Schriften entstehen würden. So gibt es die Computer-Modern-Familie einmal in der ursprünglichen, „Old TeX"-Codierung, genannt `OT1`, dann aber auch als DC-Fonts in der neuen Codierung, genannt `T1` (oder Cork- oder EC-Codierung). Auch die Mathematik-Schriften folgen verschiedenen Codierungen. Falls die Codierung nicht näher bekannt ist (dann ist die Zuordnung der Zeichen Sache des Anwenders), heißt der Typ `U` wie „unknown".

Die Codierung einer Schrift kommt zu den traditionellen Attributen wie Familie, Größe, Breite, Gewicht und Form hinzu. Dieser Aspekt wurde in den Release 2 von NFSS, kurz: NFSS2, eingearbeitet. Zur Unterscheidung, und um hervorzuheben, daß das alte NFSS nicht mehr länger

unterstützt wird, haben sich für den Benutzer die Kommandos geändert. Das ist bedauerlich für alle, die schon fleißig Gebrauch von NFSS1 gemacht haben.[4] Andererseits wird sich in den nächsten LaTeX-Versionen an der jetzigen Form der Kommandos nichts mehr ändern. Dort ersetzt NFSS2 den alten Mechanismus und wird zum Standard (seit LaTeX 2_ε).

6.4 Umschaltkommandos für die Schriftattribute

Es gibt nach wie vor die gewohnten Kommandos `\it`, `\bf`, `\tt`, `\small` usw. Zusätzlich bietet NFSS für jedes Attribut ein eigenes Umschaltkommando, bis auf Breite und Gewicht, die zusammengefaßt sind, da sie oft gemeinsam auftreten. Die Zuordnung der Attribute zu den Parametern von NFSS lautet:



Im folgenden wird immer wieder auf die Attribute in NFSS-Sprechweise eingegangen. Dabei verwende ich im Deutschen für „series“ den nicht ganz passenden, aber eingängigen Begriff „Serie“. Allgemein möchte ich verschiedene Schriftausprägungen weiterhin als „Varianten“ bezeichnen.

Die Umschaltkommandos von NFSS lauten im einzelnen:

`\fontencoding{`*Codierungskürzel*`}`
`\fontfamily{`*Familienkürzel*`}`
`\fontseries{`*Serienkürzel*`}`
`\fontshape{`*Formkürzel*`}`
`\fontsize{`*Schriftgröße*`}{`*Zeilenabstand*`}`
`\selectfont`

wobei `\selectfont` das eigentliche Umschaltkommando ist, ohne das die anderen Anweisungen wirkungslos bleiben.

Es gibt eine Abkürzung, um alle Schriftattribute bis auf die Größe umzuschalten. Sie lautet

`\usefont{`*Codierung*`}{`*Familie*`}{`*Variante*`}{`*Form*`}`

Alle Kommandos außer `\fontsize` nehmen Kürzel als Argumente. Der Rest dieses Abschnittes listet die möglichen Kürzel auf.

[4] Am Ende des Kapitels folgt ein Abschnitt, was sich bei NFSS2 geändert hat.

Codierungskürzel

Codierungskürzel gibt es nur wenige. Zunächst wurden die Codierungen der bereits bestehenden Schriften benannt: `OT1`, die „alte“ TEX-Codierung, entspricht der Codierung der Schrift *Computer Modern Roman*. `OT2` steht für die Codierung der kyrillischen CM-Schriften. Für den Mathematikssatz wurden die Namen `OML` (*Letters*, also der Text; Familie `cmm`), `OMS` (Symbols, Familie `cmsy`) und `OMX` (Extended Symbols, Famlie `cmex`) vergeben.

`T1` ist die „neue“ TEX-Codierung, auch EC-Codierung oder Cork-Codierung genannt (nach der Stadt, in der die Tagung der TEX Users Group stattfandt, auf der diese Codierung erarbeitet wurde). `U` bezeichnet eine unbekannte, schrifteigene Codierung.

Man kann ohne weiteres neue Codierungen einführen, aber solange sie nur lokal gebraucht werden, sollten sie mit dem Buchstaben `L` beginnen.

Familienkürzel

Der Mindestsatz an definierten Familienkürzeln besteht aus `cmr`, `cmss` und `cmtt` für die drei CM-Schriftfamilien mit (*Roman*) und ohne Serifen bzw. im Schreibmaschinenstil.

Prinzipiell dürfen Familienkürzel bis zu fünf Zeichen lang sein (denn drei werden für das Codierungskürzel gebraucht). Bei der NFSS-Distribution werden etliche weitere Familien unterstützt: die Schriftfamilie *Concrete* aus Knuths Buch „Concrete Mathematics“, die Familie *Pandora* von N.N. Billawala, die $\mathcal{AMS}$-Fonts der American Mathematical Society (im wesentlichen die Euler-Schriften und Erweiterungen der CM-Schriften), die „old german“ genannten Schriften von Yannis Haralambous (meistens in einem Verzeichnis namens `gothic` auf FTP-Servern zu finden) und die Standard-PostScript-Schriften von AvantGarde bis ZapfDingbats. Dazu kommen noch Symbol-Zeichensätze (z.B. für den Mathematiksatz). Später wird erklärt, wie man neue Schriften hinzufügt. Tabelle 6.1 zeigt die standardmäßig unterstützten Schriftfamilien mit ihren Kürzeln.

Alle Schriften in dieser Tabelle bis auf *Computer Modern Fibonacci*, die es nur als 8 pt-Schrift gibt, sind in einer Größe von 10 pt gesetzt (in der DIN-A4-Version des Dokumentes). Wie Sie sehen, fallen Schriften gleicher Designgröße durchaus unterschiedlich groß aus.

Serien- und Formkürzel

Formkürzel sind ein oder zwei Zeichen lang, *Serienkürzel* bis zu vier. Sie sind in Tabelle 6.2 aufgeführt. Die gängigen Kürzel sind unterstrichen. Wenn Sie Gewicht und Breite zur *Serie* kombinieren, nehmen Sie erst das Kürzel für das Gewicht, dann die Breite. ‘`m`’-Kürzel werden weggelassen, es sei denn, sowohl Gewicht als auch Breite sind normal, dann schreibt man ein einziges `m`.

Familie		Schriftprobe
cmr		Computer Modern Roman
cmss		Computer Modern Sans Serif
cmtt		Computer Modern Typewriter
cmdh		Computer Modern Dunhill
cmfr		Computer Modern Funny Roman
cmfib		Computer Modern Fibonacci
ccr		Concrete Roman
panr		Pandora Roman
euf	AMS	Euler Fraktur
eur	AMS	Euler Roman
eus	AMS	EULER SCRIPT
yfrak	Y	Yannis Fraktur
ygoth	Y	Yannis Gotilch
yswab	Y	Yannis Alte Schwabacher
yinit	Y	
pag	PS	Avantgarde
pbk	PS	Bookman
pcr	PS	Courier
phv	PS	Helvetica
pnc	PS	New Century Schoolbook
ppl	PS	Palatino-Roman
ptm	PS	Times-Roman
pzc	PS	*Zapf Chancery Medium Italic*

Tabelle 6.1: *Schriftfamilien und ihre Kürzel.*

Serie				Form	
Gewicht		Breite			
ul	ultra light	uc	ultra condensed	n	normal
el	extra light	ec	extra condensed	it	italics, kursiv
l	light	c	condensed	sl	slanted, geneigt
sl	semilight	sc	semicondensed	sc	small caps
m	medium	m	medium	u	unslanted italics
sb	semibold	sx	semiexpanded	In	invisible (SliTEX)
b	bold	x	expanded		
eb	extra bold	ex	extra expanded		
ub	ultra bold	ux	ultra expanded		

Tabelle 6.2: *Liste der Serien- und Formkürzel in NFSS; gängige sind unterstrichen.*

Übrigens gibt es einen feinen Unterschied zwischen *expanded* und *extended* bzw. zwischen *narrow* und *condensed*. Die Bezeichnungen *expanded* und *narrow* stehen für automatisch aus der Normalform gestreckte und gestauchte Schriften, die Bezeichungen *extended* und *condensed* werden für speziell von Hand entworfene breitere und engere Schriften verwendet. NFSS unterscheidet hier nicht.

Beispiele für verschiedene Kürzelkombinationen

In Tabelle 6.3 werden für einige Schriften aus den Familien Computer Modern und Concrete die entsprechenden Kombinationen von Kürzeln gezeigt. Die Codierung ist dabei einheitlich auf `OT1` gesetzt.

Ersatzmechanismen für fehlende Schriften

Es ist Ihnen sicher aufgefallen, daß die meisten Kombinationen der angegebenen Kürzel nicht zu vorhandenen Schriften führen. Was macht nun NFSS? Es sucht sich stattdessen automatisch die nächstbeste Schrift heraus und gibt eine Warnung. Das kann nicht nur passieren, wenn Sie versehentlich eine „falsche“, d.h. nichtexistente Schrift ausgewählt haben, sondern auch, wenn etwa eine *Sans Serif Slanted*-Variante in einer `\section`-Anweisung steht, aber in der Überschrift die Serie auf `bx` umgeschaltet wird, die es im Kombination mit der Form `sl` nicht gibt. Hier wählt NFSS die normal aufrechte Schrift *Sans Serif Bold Extended* als vernünftigen Ersatz aus.

NFSS verfolgt die Strategie, erst die unwesentlicheren Attribute Form und Serie durch die Normalwerte zu ersetzen. Reicht das nicht aus, um eine Ersatzschrift zu finden, wird die Familie auf die Standardfamilie zurückgesetzt. Erst ganz zuletzt wird die Codierung verändert.

Familie	Serie	Form	Schrift
cmr	m	n	Computer Modern Roman
cmr	m	it	*Computer Modern Italics*
cmr	m	sc	Computer Modern Small Caps
cmr	m	sl	*Computer Modern Slanted*
cmr	b	n	**CM Bold**
cmr	bx	n	**CM Bold Extended**
cmr	bx	it	***CM Bold Extended Italics***
cmr	bx	sl	***CM Bold Extended Slanted***
cmss	m	n	CM Sans Serif
cmss	m	sl	*CM Sans Serif Slanted*
cmss	sbc	n	**CM Sans Serif Semibold Condensed**
cmss	bx	n	**CM Sans Serif Bold Extended**
ccr	m	n	Concrete Roman
ccr	m	it	*Concrete Roman Italics*
ccr	m	sc	Concrete Roman Small Caps
ccr	c	sl	*Concrete Condensed Slanted*

Tabelle 6.3: *Verschiedene Schriften und ihre Kürzelkombinationen.*

So ersetzt NFSS eine *Computer Modern Light* oder *Ultra Bold* durch die Medium-Variante, was nicht weiter bemerkenswert ist.

Bedeutsamer ist folgendes: **Umschalten auf andere Familien setzt voraus, daß die Codierung stimmt!** Angenommen, die aktuelle Codierung ist `OT1`. Sie können jetzt nicht auf die Familie `eur` umschalten, ohne gleichzeitig auf die Codierung `U` zu wechseln, da die Schriftfamilie *Euler* nicht `OT1`-codiert ist. Der Versuch würde aus der implizit angeforderten Kombination `OT1eur` je nach festgelegter Standardschrift etwa `OT1cmr` machen.

Der Grund dafür, daß Familie und Codierung so eng miteinander verbunden sind, ist darin zu sehen, daß NFSS auf die Schriftbeschreibungen (`.fd`-Dateien) genau anhand dieser Kombination zugreift. Zu diesen Dateien finden Sie näheres in Abschnitt 6.7.

Zur Schriftgröße

Die alten LaTeX-Kommandos bleiben Ihnen erhalten. Aber sie schalten im Gegensatz zu früher nur noch die Schriftgröße um, es ist jetzt egal, ob Sie `\large\bf` wie früher oder `\bf\large` schreiben. Zusätzlich können Sie mit dem `\fontsize`-Kommando direkt die Schriftgröße zusammen mit dem neuen Zeilenabstand vorgeben. Wenn Sie den bisherigen Zeilenabstand beibehalten wollen, schreiben Sie

```
\fontsize{Schriftgröße}{\baselineskip}\selectfont
```

Zwei weitere Beispiele dazu:

```
\fontfamily{ppl}\fontsize{10.5pt}{12pt}\selectfont
```

Dieser Absatz ist in Palatino gesetzt, die Schriftgröße ist 10.5 pt und der Zeilenabstand 12 pt. Wie Sie sehen, können Sie bei PostScript-Fonts beliebige, stufenlose Größen angeben, denn diese Vektor-Schriften sind frei skalierbar.

```
\fontfamily{ccr}\fontsize{10.5pt}{12pt}
```

Im Gegensatz dazu gibt es diese Concrete-Schrift nur in festen Design-Größen von 5, 6, 7, 8, 9 und 10 pt. METAFONT kann zwar auch beliebige skalierte Versionen dieser Schrift herstellen, aber das bedeutet für jede Größe einen neuen METAFONT-Durchlauf, neue .pk-Dateien, weniger Plattenplatz. Also sind in der Regel nur wenige, gängige Vergrößerungsstufen vorhanden, die der `\magstep`-Reihe entsprechen. NFSS findet in der mitgelieferten `OT1ccr.fd`-Datei nur Deklarationen der Standardgrößen, beschwert sich und nimmt die nächstbeste Größe als Ersatz (hier 10.95 pt).

6.5 High-Level-Kommandos

L. Lamport entwarf LaTeX mit dem Hintergedanken, daß das logische Design eines Dokuments Vorrang vor dem optischen Design haben sollte, so daß man beim Schreiben nur angibt, ob man etwas hervorheben will, sich aber nicht in Layout-Details verliert, welche Schriftart in welcher Größe zum Hervorheben benutzt werden soll. In dem Sinne sind die gerade besprochenen Kommandos „Low-Level“, denn sie befassen sich direkt mit dem Erscheinungsbild des Textes. Während des Schreibens sollte man andere Befehle verwenden, die nur mit dem logischen Aufbau zu tun haben. Gute Typographie wird durch einfachen, schlichten Textsatz viel eher erreicht als durch überladenes, unausgeglichenes Design. Zum Hervorheben einzelner Textteile sollte eine kursive oder geneigte Schriftvariante verwendet werden, nur in Fällen, wo einzelne Wörter schnell auf der Seite gefunden werden müssen, Fettdruck (etwa in einem Glossar); Unterstreichungen oder Sperrschrift sind zu vermeiden. Ein einheitlicher Grauwert der Textblöcke ist das Ideal. TeX macht es einfach, diesen Regeln zu folgen, aber man kann (wie auch zwangsläufig in diesem Leitfaden) durch zuviel Schriftspielereien ebenso leicht Dokumente mit geringer typographischer Qualität produzieren.

Die Autoren von NFSS haben eine Alternative zu den alten „High-Level“-Kommandos wie `\em` und `\bf` eingebaut. Das ist ein Schritt, der unter den langjährigen TeX-Benutzern nicht auf Zustimmung stoßen mag. Der Sinn dieser Änderung, die womöglich irgendwann die alten Kommandos zum Verschwinden bringen wird, liegt in der Vereinheitlichung der Schreibweise für TeX-Befehle: Die meisten haben die Form `\`*kommando*`{`*parameter*`}`, die LaTeX-Umgebungen werden umrahmt durch `\begin{...}` und `\end{...}`. Nur die Schriftbefehle gelten ab der Stelle im

Text, wo sie auftreten. Vom Standpunkt des logischen Designs ist es besser, z.B. hervorgehobene Textblöcke richtig als solche zu begrenzen.

Mit den neuen Kommandos schreiben Sie jetzt, um einen Textblock ...

hervorzuheben:	`\emph{...}`
in einer anderen Familie zu setzen:	`\textrm{...}` , `\textsf{...}` , `\texttt{...}`
fett oder nicht fett zu setzen:	`\textbf{...}` und `\textmedium{...}`
in anderer Formvariante zu setzen:	`\textit{...}` , `\textsl{...}` , `\textsc{...}` , `\textnormal{...}`

Die meisten Kommandos sind aus den alten durch Voranstellen von `text` im Namen hervorgegangen.

6.6 Mathematiksatz und Symbole

Bislang war es in TEX und LATEX kein Problem, innerhalb von Formeln Umschaltkommandos wie `\rm` zu benutzen. Jetzt ist der Formelsatz von dem Textsatz getrennt. Es gibt nunmehr die *math version*, mit der die Erscheinung des gesamten Formelbildes verändert wird. Im Augenblick kann man zwischen normal und fett wählen, entweder wie gewohnt mit `\unboldmath` und `\boldmath`, oder besser mit

`\mathversion{normal}` $a^2 + b^2 = c^2$ bzw.
`\mathversion{bold}` $\boldsymbol{a^2 + b^2 = c^2}$

da es noch weitere Versionen gibt, etwa eine namens `euler` im Zusammenhang mit dem Style `nfconcr`.

Text in Formeln wird nicht als gewöhnlicher Text, sondern als *math alphabet* betrachtet. Gewöhnlicher Text ist verboten, d.h. ein `\rm` in einer Formel führt zu einer Fehlermeldung. Der richtige Weg ist, eines der vordefinierten Mathematik-Alphabete zu benutzen, wobei die Kommandos ganz ähnlich wie die „High-Level"-Kommandos im Textsatz lauten:

`\mathnormal{...}` , `\mathcal{...}` , `\mathrm{...}` ,
`\mathbf{...}` , `\mathsf{...}` , `\mathit{...}` ,
`\mathtt{...}` .

Besonders zu beachten ist, daß selbst das Kommando `\cal` jetzt konsequenterweise durch `\mathcal` ersetzt werden muß!

Oft kommt die Frage auf, wie man die großen Symbolzeichen wie das Integral oder das Summenzeichen mitskalieren kann. Hierzu ist der Style `nfexscal` gedacht, der die Symbole in den üblichen `magstep`-Größen und in 8 und 9 pt anbietet:

$$\Sigma\,\Sigma\,\Sigma\,\Sigma\,\Sigma\,\Sigma\,\Sigma\,\Sigma\,\Sigma$$

Zusätzlich zu `\oldstyle` gibt es ein neues Kommando nach der Art der „High-Level"-Kommandos, nämlich `\oldstylenums....`

Wenn Sie ein ganz bestimmtes Zeichen aus dem gewählten Zeichensatz brauchen, verwenden Sie den Befehl

`\symbol{`*Codenummer*`}`

Das Ergebnis ist abhängig von der Schrift und vom aktuellen Codierungsschema. Dieser Befehl ist nützlich für Symbolschriften. Ein eigener Style namens `nfpi` erlaubt den Zugriff auf die PostScript-Symbolschriften *Symbol* und *ZapfDingbats*. Er stellt u.a. folgende Kommandos zur Verfügung:

✗ die `dinglist`- und `dingautolist`-Umgebung, bei der Sie ein Zeichen aus den DingBats für die Aufzählung angeben können. Diese Liste entstand mit
```
\begin{dinglist}{55}
\item ...
\end{dinglist}
```

✗ das Kommando `\dingfill`, das genau wie die `\leaders` von TEX arbeitet, siehe Beispiel unten;

✗ das Kommando `\Pisymbol` mit zwei Argumenten, das erste legt die Familie fest (`psy` oder `pzd`), das zweite die Nummer des Zeichens, z.B. `\Pisymbol{pzd}{52}` : ✔

Die Zeichenbelegung der Schriften finden Sie im Anhang in den Tabellen B.5 und B.6. Eine andere Möglichkeit ist

`tex testfont` bzw. `plaintex testfont`

wobei Sie als Namen der Schrift entweder `rpsyr` oder `rpzdr` eingeben. Mit

`\table\bye`

wird eine Zeichensatztabelle erstellt und TEX wieder verlassen. So oder so werden Sie herausfinden, daß etwa das Scherensymbol in *ZapfDingbats* auf Position 34 liegt...

`\hbox to\textwidth{\dingfill{34}}`
✂✂✂✂✂✂✂✂✂✂✂✂✂✂✂✂✂✂✂✂✂✂✂✂✂✂✂✂✂✂✂✂

6.7 Neue Schriften

Wie sollen neue Schriften in das System aufgenommen werden? Es wird eine neue Familie und darin einige mögliche Attributkombinationen geben. Jeder erlaubten Kombination muß eine Schrift zugeordnet werden. Die Zuordnung sollte im Gegensatz zu dem alten Mechanismus von LATEX flexibel bleiben und findet daher am besten über Dateien statt. MS-DOS begrenzt die

Anzahl der verwendbaren Zeichen auf acht; man kann also kaum die Unmenge der existierenden Schriften in allen Varianten systematisch auf einzelne Dateinamen abbilden.

Daher benutzt NFSS nur Familie und Codierung für den Zugriff auf die Schriftinformation, d.h. auf die erforderliche `.tfm`-Datei. Davon ausgehend bildet NFSS einen Dateinamen, z.B. `OT1cmr.fd` für die übliche, alte CM-Roman-Schrift. Jede dieser `.fd`-Dateien (*font description*) enthält vor allem eine Reihe von `\DeclareFontShape`-Befehlen, die bekanntgeben, welche Schriftvarianten in welchen Größen existieren und wie die zugehörigen `.tfm`-Dateien heißen.

Das bedeutet, daß neue Schriften nicht direkt durch Hinzufügen einzelner Dateien zugänglich werden. Wenn Schriftvarianten zu einer bestehenden Familie/Codierung hinzukommen, muß die entsprechende `.fd`-Datei erweitert werden. Für neue Familien oder neue Codierungen müssen neue `.fd`-Dateien erstellen werden.

Diese `.fd`-Dateien werden während des TEX-Laufes beim ersten Zugriff auf die Kombination aus Familie und Codierung wie andere TEX-Dateien eingelesen und stehen daher im Arbeitsverzeichnis oder im Verzeichnis `EMTEX\TEXINPUT` unter DOS bzw. in einem Unterverzeichnis von `inputs` oder `macros` unter Unix.

Ein 'Concretes' Beispiel

Als erstes Beispiel betrachten wir die Datei `OT1ccr.fd`. Die Concrete-Schriften umfassen eine normale Variante in den Design-Größen 5–10 pt, eine kursive Variante und Kapitälchen jeweils in 10 pt Größe und eine geneigte enge Variante (*slanted condensed*) in 9 pt. Die `.tfm`-Dateien, die die Abmessungen der Zeichen für TEX enthalten, heißen `ccr5`, `ccr6`, `ccr7`, `ccr8`, `ccr9`, `ccr10`, `ccti10`, `cccsc10` und `ccslc9`.

Die `.fd`-Datei sieht im wesentlichen so aus (lassen Sie sich nicht durch Zeilenumbrüche irritieren, die sind hier bedeutungslos):

```
\DeclareFontFamily{OT1}{ccr}{}{}
\DeclareFontShape{OT1}{ccr}{m}{n}{%
  <5> <6> <7> <8> <9> <10> gen * ccr
  <10.95> <12> <14.4> <17.28> <20.74> <24.88> ccr10
}{}
\DeclareFontShape{OT1}{ccr}{m}{it}{
  <10> <10.95> <12> ccti10
}{}
\DeclareFontShape{OT1}{ccr}{m}{sc}{
  <10> <10.95> <12> cccsc10
}{}
\DeclareFontShape{OT1}{ccr}{c}{sl}{<9> ccslc9}{}
```

Die erste Anweisung teilt NFSS mit, daß diese neue Familie `ccr` in der vordefinierten OT1-Codierung (die der CM-Fonts) verfügbar ist. Dann folgen `\DeclareFontShape`-Anweisungen für die vier Varianten. Das Schema für dieses Kommando ist

\DeclareFontShape{*Codierung*}{*Familie*}{*Variante*}
{*Form*}{*Lade-Info*}{*Lade-Optionen*}

Die ersten vier Parameter sind klar. Für die meisten neuen Fonts wird man die Codierung U (unknown) wählen, da sie weder vollständig OT1- noch T1-codiert sein werden. Der letzte Parameter (*Lade-Optionen*) enthält besondere Anweisungen, die beim ersten Laden dieses *Font Shapes* durchgeführt werden sollen, und ist normalerweise leer. Die wichtigen Informationen enthält der Parameter *Lade-Info*, den wir uns näher ansehen müssen.

Betrachten wir die letzte Anweisung des Beispiels oben zuerst, da sie am einfachsten ist. Die *Lade-Info* lautet nur '<9> ccslc9'. Das bedeutet, es gibt die Schrift nur in Größe 9 (pt), und die zugehörige .tfm-Datei heißt ccslc9.tfm.

Entsprechend sagen die Deklarationen für Italics und Kapitälchen, daß es die Schrift in den Größen 10, 10.95 und 12 pt gibt, daß aber für alle drei Größen die Datei ccti10.tfm bzw. ccsc10.tfm zuständig ist (also werden die größeren Schriften aus der 10 pt-Version heraus skaliert).

Die normale Variante gibt es also in den Größen 5–10 pt und darüber hinaus in den \magstep-Schritten. Welche .tfm-Datei hier jeweils zuständig ist, hätte als Aufzählung angegeben werden können, etwa so:

```
<5> ccr5 <6> ccr6 <7> ccr7 <8> ccr8 <9> ccr9
<10> <10.95> <12> <14.4> <17.28> <20.74> <24.88>ccr10
```

Daß für die 5–10 pt-Schriften der .tfm-Name aus dem Namen 'ccr' und der Punktgröße „generiert“ wird, kann einfacher wie folgt formuliert werden:

```
<5> <6> <7> <8> <9> <10> gen * ccr
```

Anwendung: Das METAFONT-Logo

Knuth hat für das METAFONT-Logo eine einfache Schrift mit den sieben verschiedenen Buchstaben des Programmnamens entworfen. Wie würde man diese Schrift in einem Text verwenden? Die primitivste Methode besteht darin, den Font mit TEX- oder LATEX-Kommandos zu laden:

```
\font\mflogo=logo10  oder
\newfont{\mflogo}{logo10}
```

Danach kann man mit \mflogo auf die Schrift umschalten. Aber die Schrift ist festgelegt auf Größe und Form. Würde sie in einer Fußnote oder einem \section-Kommando erscheinen, wäre sie nicht so klein wie der Fußnotentext bzw. nicht so groß und fett wie der Rest der Überschrift.

Genau dafür ist NFSS ja gut. Als Codierung nehmen wir U, da nur sieben Zeichen des Zeichenvorrats belegt sind, und die Familie soll logo heißen. Also schreiben wir eine Datei Ulogo.fd,

die NFSS sagt, welche Formen und Größen dieser Schrift existieren. Fünf .tfm-Dateien stehen uns zur Verfügung, nämlich logo8.tfm, logo9.tfm, logo10.tfm, logobf10.tfm und logosl10.tfm. Die passende .fd-Datei müßte lauten:

```
% Ulogo.fd
\DeclareFontFamily{U}{logo}{}
\DeclareFontShape{U}{logo}{m}{n}{
 <8> <9> gen * logo
 <5> <6> <7> <10> <10.95> <12>
 <14.4> <17.28> <20.74> <24.88> logo10
 }{}
\DeclareFontShape{U}{logo}{bx}{n}{
 <5> <6> <7> <8> <9> <10> <10.95> <12>
 <14.4> <17.28> <20.74> <24.88> logobf10
 }{}
\DeclareFontShape{U}{logo}{m}{sl}{
 <5> <6> <7> <8> <9> <10> <10.95> <12>
 <14.4> <17.28> <20.74> <24.88> logosl10
 }{}
\DeclareFontShape{U}{logo}{m}{it}{
 <-> sub * logo/m/sl
 }{}
```

Da habe ich noch eine neue Anweisung 'sub' (substitute) eingeschmuggelt: Die Italics-Variante, die es nicht gibt, aber auf die vielleicht zugegriffen wird, wird auf die Slanted-Variante zurückgeführt. Dabei steht <-> für alle Größen.

Diese .fd-Datei kommt in ein Eingabe-Verzeichnis für TEX, z.B. in das Arbeitsverzeichnis, und dann kann man sich ein Makro

```
\def\MF{{\fontencoding{U}\fontfamily{logo}%
\selectfont METAFONT}}
```

definieren und überall \MF benutzen (beachten Sie die gleichzeitige Umschaltung der Codierung). Es sind alle gängigen Größen abgedeckt, also paßt sich die Schrift auch innerhalb von Fußnoten an; für Überschriften gibt es eine fette Variante, für Kopfzeilen eine geneigte. Man muß nur bei der Definition des Makros aufpassen, daß Größe, Form und Variante beibehalten werden, sonst funktioniert es nicht!

Neue PostScript-Schriften einzurichten ist eine größere Sache (es sei denn, Sie haben bereits fertige .tfm- und .vf-Dateien, dann brauchen Sie nur noch die .fd-Datei zu schreiben). Wir sehen uns das genauer im Kapitel 7 über virtuelle Fonts an.

Kommando	Anfangswert
\encodingdefault	OT1
\familydefault	\rmdefault
\seriesdefault	m
\shapedefault	n
\rmdefault	cmr
\sfdefault	cmss
\ttdefault	cmtt
\bfdefault	bx
\mddefault	m
\itdefault	it
\sldefault	sl
\scdefault	sc
\normalshapedefault	n

Tabelle 6.4: *Liste der Font Hooks von NFSS und deren Voreinstellungen. Achtung: In NFSS2 hieß* `\mddefault` *noch* `\mediumseriesdefault`.

6.8 Font Hooks

Noch wissen Sie nicht, wie man die Schriftfamilie für ein gesamtes Dokument verändern kann. Sie können zwar die Familie für den Text ändern, aber die Überschriften usw. bleiben in Computer Modern Roman, denn in den Definitionen z.B. des `\section`-Kommandos stehen Befehle wie `\rm`, die auf `cmr` schalten. An dieser Stelle können Sie aber durch die sogenannten „Font Hooks“, Haken, eingreifen. z.B. bestimmt das Kommando `\familydefault` die Standard-Schriftfamilie, so daß

```
\renewcommand{\familydefault}{\sfdefault}
```

am Anfang eines Textes den CM Sans Serif-Font als Grundschrift einstellt.[5] Es gibt aber noch mehr solche Hooks, vgl. Tabelle 6.4.

Mit diesen Font Hooks können Sie auch für das gesamte Dokument PostScript-Schriften zur Grundlage machen. Dabei ist zu beachten, daß die fetten Versionen der PostScript-Schriften nicht gleichzeitig breit sind, also die Serie nicht auf `bx`, sondern nur auf `b` umgeschaltet werden muß, eine Aufgabe für den Hook `\bfdefault`, der normalerweise auf `bx` steht. Außerdem möchten Sie nicht nur die `\rm`-Schrift umstellen, sondern auch die `\sf`-Schrift und die `\tt`-Schrift, wobei es aber im Gegensatz zur Computer Modern-Familie keine Standardschrift gibt, die mit und ohne Serifen und als Typewriter vorliegt. Es müssen also verschiedene PostScript-Familien benutzt werden.

[5] Fragen Sie mich aber bitte nicht, wieso manche Seiten- und Fußnotennummern immer noch in Computer Modern Roman gesetzt werden.

Eine vollständige Umschaltung auf Times-Roman (mit Helvetica als Sans Serif-Schrift und Courier als Typewriter) sollte so aussehen:

```
\renewcommand{\rmdefault}{ptm}
\renewcommand{\sfdefault}{phv}
\renewcommand{\ttdefault}{pcr}
\def\bfdefault{b}
```

Genau das (bzw. Analoges) geschieht in den mitgelieferten Styles

`nfavant`, `nfbookm`, `nfhelve`, `nfnewce`, `nfpalat` und `nftimes`.

(Wo wir gerade dabei sind: Es gibt auch Styles für die anderen Familien wie Concrete, Pandora und Euler, sie lauten `nfconcr` bzw. `nfbeton`, `nfpandor` und `nfeuler`.)

Diese Styles nehmen keinen Einfluß auf die Codierung, Sie können also immer noch frei zwischen `OT1` und `T1` wählen, wobei Vor- und Nachteile der `T1`-Codierung im folgenden Abschnitt erörtert werden.

6.9 Die DC-Fonts und Umlaute

DC-Fonts sind nichts anderes als erweiterte CM-Fonts, die anders (nämlich `T1`-) codiert sind. Das Besondere an der `T1`-Codierung ist, daß sehr viele internationale Sonderzeichen als eigenständige Zeichen enthalten sind. Gleichzeitig gibt es eine neue deutsche Trenntabelle, die jetzt mit Umlauten ausgestattet ist, so daß Wörter mit Umlauten ganz normal mitgetrennt werden. Nun könnten Sie meinen, Sie brauchten nur den Hook `\encodingdefault` auf `T1` zu setzen, und schon hätten Sie ein Dokument aus DC-Fonts. So einfach ist es aber nicht, denn die Mathematik-Befehle müssen auch noch angepaßt werden. Da sind die Mathematik-Alphabete umzusetzen, die griechischen Großbuchstaben können jetzt nicht mehr aus dem `cmr`-Font in der neuen Codierung genommen werden, sondern müssen von Hand auf die alte Codierung umgesetzt werden, und die Akzente sind anders codiert. Diese Umsetzungen erledigt für Sie der mitgelieferte Style `nfdcfnt.sty`, den Sie in der Anweisung `\documentstyle` mitangeben.

Doch schon bei der Eingabe der Umlaute gibt es ein Problem: Unter MS-DOS liegen die deutschen Umlaute woanders als beim Standard ISO-8859-1 (Latin-1), an den sich z.B. das X Window System, MS Windows und AmigaOS halten (siehe Tabelle). Das gleiche gilt auch für andere nationale Sonderzeichen, die beim PC sogar noch je nach Codepage (437 oder 850) verschiedene Positionen einnehmen.

Codierung	Ä	Ö	Ü	ä	ö	ü	ß
MS-DOS (850)	142	153	154	132	148	129	225
ISO Latin-1	196	214	220	228	246	252	223
T1/EC	196	214	220	228	246	252	*255*

ISO- und MS-DOS-Zeichensätze sind Erweiterungen des ASCII-Zeichensatzes auf 256 Zeichen. Dabei ist der ISO-Zeichensatz erheblich systematischer angelegt als die Codepage 850 von MS-DOS. Im ISO-Zeichensatz ist die Belegung in 32er-Gruppen organisiert: die Positionen 128–159 sind unbelegt, von 160 bis 191 stehen verschiedene andere Symbole, ab 192–255 sind die üblichen (west-)europäischen Sonderzeichen untergebracht, wobei die ersten 32 davon die Großbuchstaben, die zweiten die Kleinbuchstaben sind. Das 'ß', welches keinen Großbuchstaben besitzt, ist die einzige Ausnahme von der Regel.

Die Codierung der DC-Fonts ist an ISO Latin-1 angelehnt. Im noch freien Bereich 128–159 wurden einige der seltener gebrauchten akzentuierten Zeichen untergebracht. Da die großen und kleinen Umlaute genau um 32 auseinanderliegen, wie es bei den gewöhnlichen Buchstaben auch ist, wurde das 'ß' in einer großen Version (SS) auf 223 gelegt, und das richtige, kleine 'ß' auf 255!

Schreibt man nun eine TEX-Datei auf einem ISO Latin-1 System und verwendet die DC-Fonts, erscheint das 'ß' als 'SS'. Diese ärgerliche Nebensache kann aber leicht durch die Konstruktion

```
\catcode`\^^df=\active \def^^df{"s}
```

umgangen werden, sofern der `german.sty` oder der `german3.sty` verwendet wird. Andere Lösungen, die direkt das richtige Zeichen auf die Position 223 definieren, scheitern am Umsetzen in die großgeschriebene Version in Kopfzeilen, wo dann ein kleines ß unter all den Großbuchstaben steht.

Schreiben Sie aber unter MS-DOS mit den IBM-Umlauten der Codepage 437 oder 850, gibt es auch für Sie eine Lösung, sofern Sie emTEX benutzen: Man verwendet eine `.tcp`-Datei (tex codepage), die die IBM-Umlaute soweit wie möglich auf ISO Latin-1 umsetzt. So können Sie Ihre TEX-Datei „normal“ eintippen. Nur beim Austausch mit ISO Latin-1 Systemen müssen Sie die Umlaute erst umwandeln, entweder in die ISO-Codes oder in die Standard-Notation von TEX.

Die Umcodierung von PostScript-Schriften, die wieder eine andere Zeichensatzbelegung haben, normalerweise die *Adobe Standard Encoding*, ist ein Thema für das nächste Kapitel.

Schließlich will ich noch darauf hinweisen, daß man auch ohne die DC-Fonts die Umlaute direkt eingeben kann. Für emTEX-Benutzer sowieso ein alter Hut, denn dort gibt es ja die `.tcp`-Dateien, die die Umlaute in TEX-Befehle verwandeln können. Es gibt aber eine zweite Möglichkeit, die auch das Transportieren von TEX-Dateien mit IBM- oder ISO-Umlauten erleichtert: Man kann jeden Umlaut zu einem aktiven Zeichen machen und dieses Einzeichen-Kommando entsprechend auf TEX-gerechte Umlaute definieren. Das heißt, man würde eine Datei `doscodes.sty` erzeugen, die z.B. die Zeilen

```
\catcode`\^^84=\active \def^^84{"a}
\catcode`\^^e1=\active \def^^e1{"s}
...
```

enthält, und eine weitere Datei `isocodes.sty` mit den Anweisungen

```
\catcode`\^^e4=\active \def^^e4{"a}
\catcode`\^^df=\active \def^^df{"s}
...
```

So müßte man nur den entsprechenden `...codes`-Style mit angeben und wäre unabhängig vom verwendeten Betriebssystem. Allerdings würde man die Dateien auf Fremdsystemen nicht vernünftig bearbeiten können, da der Text unleserlich wird. Außerdem wäre es nicht möglich, die Umlaute richtig zu trennen.

Abschließend sei noch erwähnt, daß es bei Ataris TOS zwei gleich aussehende Zeichen für ß und β gibt, wobei leider das Atari-β mit dem DOS-ß übereinstimmt; die Taste 'ß' der deutschen Tastatur erzeugt aber den Code 158...

6.10 Änderungen gegenüber LaTeX ohne NFSS und gegenüber NFSS1

Hier fasse ich die wichtigsten Fehlerursachen und sonstigen Unterschiede zusammen.

Unterschiede zu LaTeX ohne NFSS

- ✯ Kommandos wie `\bf fett und \it italics` führen unter NFSS nicht mehr zu **fett und** *italics*, sondern zu **fett und *italics***.
- ✯ Innerhalb von Formeln sind `\rm`, `\bf`, `\cal` durch `\mathrm...` etc. zu ersetzen — gilt nicht für LaTeX 2_ε!
- ✯ Mit NFSS gibt es keine „festverdrahteten" Fontkommandos mehr. Das betrifft Befehle wie `\tenrm` oder `\fivesy`, die zuhauf in der alten Datei `lfonts.tex` definiert wurden. Diese Befehle waren eigentlich auch nur interne Befehle von LaTeX, aber leider benutzen manche Zusatzpakete wie PicTeX und GNUplot diese Befehle noch. Abhilfe: Die fehlenden Befehle passend definieren, etwa

```
\def\tenrm{\usefont{OT1}{cmr}{m}{n}%
\fontsize{10pt}{\baselineskip}\selectfont}
```

Unterschiede zu NFSS, Release 1

- ✯ Codierungs-Schemata sind neu.
- ✯ Die Kommandos `\family`, `series`, `\shape` und `\size` wurden durch `\fontfamily` etc. ersetzt.

- Die Mathe-Alphabete `\mit` und `\cal` wurden ersetzt durch `\mathnormal` und `\mathcal`.
- Die ganzen internen Kommandos wie `\declare@font`, `\extra@defs`, `\new@fontshape`, `\subst@fontshape`, `\newmathalphabet` und `\addtoversion` gibt es nicht mehr (sind durch ein vernünftiges User-Interface ersetzt worden).
- Alle NFSS1-Style-Dateien geben jetzt Warnmeldungen und laden die richtigen NFSS2-Styles.
- Bei NFSS1 führte der Befehl `\size{14}{16pt}` dazu, daß die Schriftgröße 14.4 pt geladen wurde, denn die Angabe '14' war nur ein Label, keine wirkliche Größenangabe. Jetzt wird die genaue Schriftgröße geladen (bzw. die nächstbeste Ersatzgröße).
- Die ganzen Style-Dateien für PostScript-Schriften und auch für die Concrete-Schriften enthalten `\declare@font`-Befehle und funktionieren nicht mehr. Die neuen Styles beginnen alle mit '`nf`', z.B.
  `nftimes.sty`.

Kapitel 7

Virtuelle Fonts und PostScript-Schriften

Virtuelle Fonts sind Schriften, die aus einer oder mehreren gewöhnlichen Schriften abgeleitet wurden. Dabei wird für jedes Zeichen des virtuellen Fonts angegeben, welches Zeichen aus einer gewöhnlichen Schrift verwendet werden soll; es können auch mehrere Zeichen zu einem neuen zusammengesetzt werden (z.B. Akzentzeichen mit Buchstaben). Der Haupteinsatzbereich von virtuellen Schriften sind PostScript-Schriften. Warum?

Ein PostScript-Drucker kennt bereits etliche Schriften von sich aus, sogenannte *residente Schriften*. Um diese verwenden zu können, haben wir drei Probleme zu lösen,

1. TEX muß wissen, wie groß die einzelnen Zeichen sind, um Sie in Boxes verpacken und nebeneinander stellen zu können;

2. die Codierungsschemata, von denen TEX ausgeht, und die Zeichenpositionen in der residenten Schrift müssen nicht übereinstimmen, bedürfen also ggf. einer Umcodierung;

3. der dvi-Treiber (für uns dvips) muß wissen, daß eine bestimmte Schrift nicht als `.pk`-Datei vorliegt, sondern schon dem Drucker bekannt ist.

7.1 Problemstellung und Lösungsweg

Zuerst müssen wir also eine `.tfm`-Datei bereitstellen, in der die Abmessungen der Zeichen und andere Informationen über Ligaturen[1] und über Unterschneidungen[2] (Kerning) enthalten sind. Zu PostScript-Schriften gibt es frei verfügbar ähnliche Metrik-Dateien, die *Adobe font metrics* (`.afm`). Ein Programm namens afm2tfm, das zu dvips gehört, besorgt die Erzeugung von `.tfm`-Dateien für TEX aus `.afm`-Dateien.

[1] aus mehreren Buchstaben zusammengesetzte Zeichen, vergleiche z.B. ffi / ffi.

[2] Unterschneiden von aufeinanderfolgenden Zeichen, die sonst optisch zu weit getrennt stehen würden, etwa AV statt AV.

Das zweite Problem wird so gelöst, daß zu jeder residenten Schrift eine Virtuelle-Font-Datei, eine `.vf`-Datei, angelegt wird. Diese Datei enthält jegliche Umcodierung. Sie kann sogar neue Zeichen aus anderen Zeichen zusammensetzen: wenn z.B. der residente Font kein Ä hat, aber ein A und die diaresis (¨), so werden die beiden gemäß den Anweisungen der `.vf`-Datei übereinander gestapelt.

Das dritte Problem wird dadurch gelöst, daß dvips über eine Tabelle der vorhandenen PostScript-Schriften verfügt, die Datei `psfonts.map`. Dabei spielt es keine Rolle, ob diese Schriften resident im Drucker sind oder als Schriftdateien vorliegen. Sobald dvips auf einen Schriftnamen trifft, der in dieser Tabelle aufgeführt ist, sucht es nicht nach einer passenden `.pk`-Datei.

Verfolgen wir nun den Weg vom Font-Befehl in TEX bis zur fertigen PostScript-Datei!

Der Treiber-Mechanismus bei externen Schriften

Angenommen, die aktuelle Schriftgröße ist 10 pt, und wir geben das Fontkommando

```
\usefont{T1}{cmr}{m}{it} .
```

Wenn TEX diesen Befehl liest, schauen die NFSS2-Routinen in der (vielleicht schon geladenen) Datei `T1cmr.fd` nach, ob es diese Variante gibt, und sie finden heraus, daß die Datei mit den Schriftabmessungen `dcti10.tfm` heißt. TEX schreibt also diesen Namen `dcti10` in die `.dvi`-Datei hinein.

Wenn dvips diese `.dvi`-Datei bearbeitet, findet es diese Eintragung und überprüft zuerst, ob es sich um einen virtuellen Font handelt: Es sucht nach der Datei `dcti10.vf`, die es aber nicht gibt! Dann durchsucht es die Einträge in der Datei `psfonts.map`, ob die Schrift etwa eine residente PostScript-Schrift ist — erfolglos. Also ist es eine gewöhnliche, nicht-residente Schrift, und dvips muß dem Drucker mitteilen, wie die Schrift aussieht. Ein Punkteraster aller Zeichen dieser Schrift steht in der Datei `dcti10.pk` (unter MS-DOS) bzw. `dcti10.300pk` (unter Unix, bei 300 dpi Basisauflösung). dvips durchsucht nun die ihm bekannten Fontlibraries (falls es welche gibt) nach dieser Datei und guckt sonst in die einschlägigen pk-Verzeichnisse. Es lädt die Datei und schreibt die Rasterinformationen in die PostScript-Datei, so daß der Drucker nunmehr die Schrift kennt.

Der Treiber-Mechanismus bei residenten Schriften

Nun nehmen wir an, wir verwenden eine PostScript-Schrift mit

```
\usefont{OT1}{ptm}{m}{n} .
```

TEX ordnet per NFSS2 die Datei `ptmr.tfm` zu (die Systematik der PostScript-Schriftdateien finden Sie in Anhang C) und verwendet die alte TEX-Codierung OT1. dvips sucht wieder zuerst nach der `.vf`-Datei (`ptmr.vf`) und wird diesmal fündig. In der `.vf`-Datei steht für jede OT1-Zeichenposition das entsprechende Zeichen aus dem passenden residenten Font (in *Adobe*

Standard Encoding). Da die OT1-Codierung sowohl „normale“ als auch griechische Buchstaben enthält, wie man z.B. an der Zeichentabelle des `cmr10`-Fonts (Tabelle B.1) ablesen kann, reicht ein einzelner PostScript-Font nicht aus, um die vollständige OT1-Codierung abzudecken. Die normalen Buchstaben werden der Schrift *Times-Roman* entnommen, die griechischen Buchstaben der Schrift *Symbol*.

Also enthält die Datei `ptmr.vf` Verweise auf die zwei Standard-PostScript-Schriften *Times-Roman* und *Symbol*. Nur sind unter MS-DOS und manchen anderen Betriebssystemen keine Namen zugelassen, die länger als acht Zeichen sind: Es muß abgekürzt werden. In Wirklichkeit stehen in der `.vf`-Datei nicht die Einträge '`Times-Roman`' oder '`Symbol`', sondern die Kurzformen '`rptmr`' und '`rpsyr`'. Das führende '`r`' bedeutet etwa soviel wie „roh“, naturbelassen, original Adobe Standard Encoding. Genaueres zu dem Abkürzungsschema finden Sie in Abschnitt C.

In seiner Tabelle `psfonts.map` findet dvips die folgenden Einträge:

```
rptmr Times-Roman
rpsyr Symbol
```

Damit weiß es, daß diese Schriften resident sind, und wie deren wirkliche Namen lauten. dvips braucht nur noch die Größe der Zeichen für die Abstände beim Positionieren. Diese steht in den entsprechenden `.tfm`-Dateien `rptmr.tfm` und `rpsyr.tfm`.

Der Treibermechanismus bei nichtresidenten PostScript-Schriften

Als Beispiel diene die Schrift *Utopia-Regular*. Die Kurzform des Namens lautet `putr`. Es existiere eine Datei `putr.pfa`, die die Schriftinformationen enthält, und eine Datei `OT1put.fd` für NFSS2.

Die Zeile

```
\usefont{OT1}{put}{m}{n}
```

führt dazu, daß TEX den Schriftnamen `putr` in der `.dvi`-Datei notiert. dvips findet dazu eine Datei `putr.vf`, deren Zeichen aus der Schrift `rputr` stammen.[3] In der Datei `psfonts.map` steht die Zeile

```
rputr Utopia-Regular <putr.pfa
```

dvips wird daraufhin die Datei `putr.pfa` mit den Schriftdefinitionen zu den anderen Header-Dateien an den Anfang der PostScript-Datei stellen. Damit ist dem PostScript-Drucker die

[3] Ihnen fällt vielleicht auf, daß ich dieses Mal nichts über die griechischen Buchstaben in der OT1-Codierung gesagt habe. Es gibt leider nicht für jede PostScript-Schrift ein passendes griechisches Alphabet. Bei den Standard-Schriften passen nur Times-Roman und Symbol zusammen; bei den kommerziell vertriebenen Lucida-Schriften gibt es eine Zusatzschrift namens Lucida-Math. Da der Mathematiksatz über das *math alphabet* gesondert behandelt wird und die griechischen Buchstaben sonst i.d.R. keine Rolle spielen, verzichtet man auf einen kompletten OT1-Zeichenvorrat, nennt die Schrift aber trotzdem OT1-codiert.

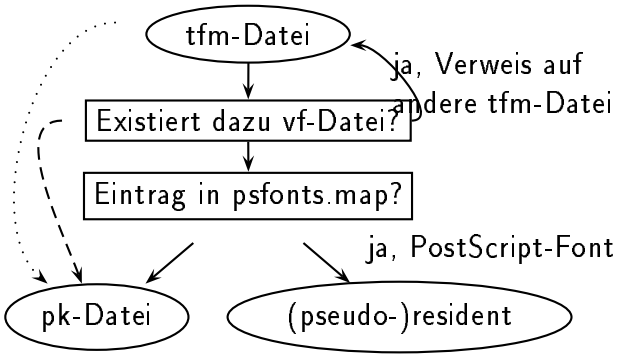


Abbildung 7.1: *Der Treiber-Mechanismus von dvips im Vergleich mit anderen dvi-Treibern. Die gepunktete Linie markiert den Ablauf in einem alten Treiber, der noch keine virtuellen Fonts kennt. Die gestrichelte Linie zeigt den Ablauf in einem Treiber wie xdvi, der zwar virtuelle Fonts, aber keine pseudoresidenten PostScript-Fonts kennt.*

Schrift *Utopia-Regular* bekanntgemacht, und dvips kann im weiteren so tun, als sei die Schrift resident.

Vorläufige Zusammenfassung

Das war ganz schön verwirrend, darum nochmal kurz und knapp:

- Es gibt zwei Arten von `.tfm`-Dateien:
  Die eine ist nach OT1 codiert und wird von TEX benutzt.
  Die andere ist nach Adobe codiert und wird von dvips benutzt.

- Die `.vf`-Datei zu einer OT1-codierten `.tfm`-Datei enthält Verweise auf Kurznamen der Adobe-codierten PostScript-Fonts.

- dvips sucht bei jedem TEX-Fontnamen erst nach der `.vf`-Datei. Findet es diese, handelt es sich um einen virtuellen Font.

- Danach sucht dvips in der Tabelle `psfonts.map`. Findet es den Kurznamen dort, wird er von dvips durch den richtigen PostScript-Namen ersetzt. Falls die Schrift nicht resident ist, wird ihre `.pfa`- oder `.pfb`-Datei der PostScript-Datei als Header vorausgeschickt.

7.2 Das Programm afm2tfm

Zum dvips-Paket gehören von Hause aus die `.tfm`-Dateien der OT1- und Adobe-codierten PostScript-Schriften. Diese `.tfm`-Dateien wurden aus den entsprechenden `.afm`-Dateien konstruiert. Bei den rohen Schriften wurde eine Eins-zu-Eins-Umsetzung vorgenommen, bei den OT1-codierten Schriften wurde zusätzlich umcodiert, bei manchen Varianten wurde eine Streckung oder Stauchung in der Breite durchgeführt, eine Neigung oder eine SmallCaps-Variante erzeugt. All das leistet das Programm afm2tfm, das hier kurz beschrieben wird; die vollständige Anleitung findet sich in der Original-Anleitung zu dvips.

7.2.1 Virtual property lists

afm2tfm wandelt `.afm`-Dateien in beliebig codierte `.tfm`-Dateien um. Wenn die Codierung unverändert bleibt, sprechen wir von „rohen“ Fontdateien. Wird die Codierung jedoch verändert, erzeugt afm2tfm eine sogenannte `.vpl`-Datei (*virtual property list*), eine lesbare und editierbare Textdatei.

In der `.vpl`-Datei sind sowohl die Zeichenzuordnung zu verschiedenen Schriften als auch die Zeichenabmessungen enthalten. Mit vptovf wird die `.vpl`-Datei in eine `.vf`-Datei für den Treiber und eine `.tfm`-Datei für TEX umgewandelt. Mit vftovp erhalten Sie umgekehrt zu der `.vf`- und `.tfm`-Datei eine `.vpl`-Datei.[4]

Wenn nur innerhalb einer Schrift umcodiert werden soll, ist afm2tfm in der Lage, das automatisch vorzunehmen. Es benötigt dazu nur den Codierungsvektor (siehe nächster Abschnitt). Sobald mehrere Schriften für einen virtuellen Font herangezogen werden, muß man die `.vpl`-Datei von Hand bearbeiten.

Als Beispiel dazu betrachten wir den Fall, daß man dem OT1-Layout entsprechend griechische Großbuchstaben in einem virtuellen *Times-Roman*-Font haben will. Die Vorarbeit leistet afm2tfm, das uns *Times-Roman* mit OT1-Codierung erzeugt. Es folgen kurze Ausschnitte aus der Datei `ptmr.vpl`:

```
(VTITLE Created by afm2tfm Times-Roman -v ptmr)
...
(CODINGSCHEME TEX TEXT + ADOBESTANDARDENCODING)
...
(MAPFONT D 0
   (FONTNAME rptmr)
   (FONTCHECKSUM 0 ...)
   )
...
(LIGTABLE
```

[4] Übrigens kann man auch zu `.tfm`-Dateien von gewöhnlichen (d.h. nicht virtuellen) Schriften eine lesbare Darstellung bekommen. Die beiden Programme dazu heißen tftopl und pltotf. '`pl`' steht für *property list*.

```
      ...
      )
   (CHARACTER O 15
      (CHARWD R 0.18)
      (CHARHT R 0.673)
      (MAP
         (SETCHAR O 251)
         )
      )
...
```

Nach einigen Zeilen mit allgemeinen Informationen folgen die dem virtuellen Font zugrundeliegenden Schriften als MAPFONT-Einträge. Bislang haben wir nur einen Font, Font Nr. 0, und zwar rptmr, also *Times-Roman* in *AdobeStandardEncoding*. Dann folgt die Tabelle für Ligaturen und Kerning (LIGTABLE). Zum Schluß kommt die Liste der Zeichen, die in dem virtuellen Font definiert sein sollen, hier beginnend mit der Zeichenposition oktal 15. Für jedes Zeichen wird außer Breite (CHARWD), Höhe (CHARHT), Tiefe (CHARDP) und Italic Correction (CHARIC) in einem MAP-Eintrag das zu verwendende Zeichen aus der Ausgangsschrift angegeben.

Die griechischen Buchstaben werden dem *Symbol*-Font entnommen, der den gleichen Stil wie *Times-Roman* aufweist. Die nötigen Informationen besorgt man sich, indem man die Datei rpsyr.tfm mit tftopl in die lesbare Version rpsyr.pl umwandelt. So erfährt man die Prüfsumme der Schrift und die Abmessungen der Zeichen.

Nun wird die Datei ptmr.vpl bearbeitet. Zuerst fügt man einen zweiten MAPFONT-Eintrag hinzu, der die Nummer Eins bekommt:

```
(MAPFONT D 1
   (FONTNAME rpsyr)
   (FONTCHECKSUM O ...)
   )
```

Strenggenommen müßten auch die Kerning-Tabellen bearbeitet werden. Wir verzichten aber darauf und tragen nur noch die elf griechischen Großbuchstaben Γ, Δ, Θ, Λ, Ξ, Π, Σ, Υ, Φ, Ψ und Ω vor der Zeichenposition oktal 15 ein. Als Muster für die Zuordnung der Zeichenpositionen kann die Zeichensatztabelle von cmr10 dienen (Tabelle B.1).

```
(CHARACTER O 0
   (CHARWD R ...)
   (CHARHT R ...)
   (CHARIC R ...)
   (MAP
      (SELECTFONT D 1)
      (SETCHAR C G)
      )
   )
```

...

Die MAP-Einträge für die Großbuchstaben beziehen sich nicht auf die Standardschrift (mit der Nummer 0), sondern auf Schrift Nr. 1, wie sie im MAPFONT-Eintrag vorher definiert wurde. An Position 0 soll das Zeichen 'Γ', das im *Symbol*-Zeichensatz an der Stelle steht, wo sonst das Zeichen 'G' ist. Statt einer oktalen Angabe (`SETCHAR O 107`) wurde hier die zeichenbezogene Angabe (`SETCHAR C G`) gewählt.

Danach kann man die bearbeitete Datei `ptmr.vpl` durch vptovf schicken und die Schriftdateien `ptmr.tfm` und `ptmr.vf` erzeugen.

7.2.2 Der Aufruf von afm2tfm

Eine veränderte Output-(TEX)-Codierung erreicht man mit der Option '-v', eine veränderte Input-(PostScript)-Codierung mit '-p', '-t' oder '-T', auf die der Name der Codierungsdatei folgt, die in PostScript-Format sein muß und dort einen Vektor mit genau 256 Einträgen definiert. Als Beispiel können Sie sich die schon erwähnte Datei `ec.enc` im `ps`-Verzeichnis ansehen. Ligaturen und Kerning-Informationen werden afm2tfm über Kommentare der Form `% LIGKERN ...` bekanntgegeben (ohne daß PostScript davon etwas merkt). Es muß der Name der Ausgabe-`.vpl`-Datei angegeben werden; dazu dient die Option '-v'.

Außer der Umcodierung können die Schriften auch transformiert werden. Sie werden geneigt mit der Option '-s' (*slant*, ein vernünftiger Wert ist ± 0.167), gestreckt mit '-e' (*expand*, hier ist 1.2 und 0.8 sinnvoll), einen SmallCaps-Font bekommt man mit '-V' statt '-v', die Höhe der Kapitälchen wird mit '-c' angegeben.

7.2.3 Beispiele

Wir gehen von der (per FTP beschaffbaren) `.afm`-Datei für *Times-Roman* aus (`ptmr.afm`) und erzeugen verschiedene virtuelle Fonts. Gezeigt werden die erforderlichen Befehle. Die Ausgaben von afm2tfm, die in die Datei `psfonts.map` einzutragen sind, sind kursiv gedruckt. Bei afm2tfm brauchen die Suffixe nicht angegeben zu werden, sind aber der Deutlichkeit halber aufgeführt.

Die rohe, also noch Adobe Standard-codierte Schrift erhalten wir mit

```
afm2tfm ptmr.afm rptmr.tfm
```

rptmr Times-Roman

Eine TEX-codierte Schrift, also den ersten virtuellen Font, gibt's mit den Anweisungen

```
afm2tfm ptmr.afm -v ptmr.vpl rptmr.tfm
vptovf ptmr.vpl ptmr.vf ptmr.tfm
```

rptmr Times-Roman

(Hier wurde die gleiche rohe Datei noch einmal erzeugt.)

Eine geneigte Version der Times, gleich richtig TEX-codiert, erhalten wir mit

```
afm2tfm ptmr.afm -s 0.167 -v ptmro.vpl rptmro.tfm
vptovf ptmro.vpl ptmro.vf ptmro.tfm
```

rptmro Times-Roman "0.167 SlantFont"

Ebenso erzeugt man gestreckte und gestauchte Schriften:

```
afm2tfm ptmr.afm -e 1.2 -v ptmrre.vpl rptmrre.tfm
vptovf ptmrre.vpl ptmrre.vf ptmrre.tfm
afm2tfm ptmr.afm -e 0.8 -v ptmrrn.vpl rptmrrn.tfm
vptovf ptmrrn.vpl ptmrrn.vf ptmrrn.tfm
```

rptmrre Times-Roman "1.2 ExtendFont"
rptmrrn Times-Roman "0.8 ExtendFont"

Diese transformierten Schriften verhalten sich wie residente Schriften, da sich ihre Erzeugung druckerintern abspielt. Die Datei `psfonts.map` enthält die Informationen, wie der Drucker diese Schriften aus den residenten erzeugen kann.

Zu einer virtuellen Schrift kann jetzt eine SmallCaps-Variante geschaffen werden. Man beachte den Unterschied: Jetzt ist der letzte Parameter für afm2tfm *kein* roher Font, sondern ein virtueller Font (der bereits bestehen kann, aber nicht muß. Er wird überschrieben). Etwa eine normale SmallCaps-Variante:

```
afm2tfm ptmr.afm -V ptmrc.vpl ptmr.tfm
```

Oder eine geneigte SmallCaps-Variante:

```
afm2tfm ptmr.afm -s 0.167 -V ptmrocr.vpl ptmro.tfm
```

7.2.4 T1-codierte PostScript-Schriften

Aus den schon angeführten Gründen empfiehlt es sich, die PostScript-Schriften auch T1-codiert verwenden zu können. Dazu muß ich vorausschicken, daß in den PostScript-Schriften normalerweise weitaus mehr Zeichen definiert sind, als sich in 256 Zeichenpositionen unterbringen ließen. Diese Zeichen, hauptsächlich nationale Sonderzeichen, sind innerhalb des Font-Dictionaries über Befehlsnamen zugänglich, z.B. produziert `/adieresis` ein 'ä'. Eine Liste dieser Namen finden Sie in Anhang B.

Leider sind diese Zeichen nicht außerhalb des Fonts direkt ansprechbar. Der einzige Weg, Zeichen auf die Druckseite zu bekommen, führt über den Operator `show` und seine Varianten, die allesamt nur Strings aus 8-Bit-Zeichen verarbeiten können. Darum gehört zu einer Schrift der sogenannte Encoding-Vektor, eine Tabelle mit 256 Einträgen, die der Zeichennummer den Befehlsnamen zuordnet. Zwei Codierungen sind in PostScript vordefiniert, die *AdobeStandardEncoding* und die *ISOLatin1Encoding*, wobei nur die erste standardmäßig verwendet wird.

Nun kann man ohne weiteres einen beliebigen neuen Encoding-Vektor definieren. Für die T1- oder EC-Codierung ist das bereits geschehen. Der Vektor nennt sich *ECEncoding* und ist zu finden in der Datei `ec.enc` im Verzeichnis

`EMTEX\PS` (bei MS-DOS) bzw.
`/usr/local/lib/texmf/dvips` (bei Unix).

Wie kommt diese Codierung nun in einen normalen PostScript-Font? — Dazu wird der ursprüngliche Font (genauer: Das Font-Dictionary) kopiert bis auf den `FID`-Eintrag (FontID), die Codierung im neuen Dictionary umdefiniert und schließlich der so entstandene neue Font fest definiert, wobei eine neue `FID` vergeben und eingetragen wird. Für die Interessierten am Beispiel *Times-Roman* (Voraussetzung: Der Vektor `ECEncoding` wurde vorher bereits definiert):

```
/Times-Roman findfont dup length dict begin
  {1 index /FID ne {def} {pop pop} ifelse} forall
  /Encoding ECEncoding def
  currentdict end
/Times-Roman-EC exch definefont pop
```

Danach kann auf die neu codierte Schrift mit

```
/Times-Roman-EC findfont
```

in gewohnter Weise zugegriffen werden.

Wie wird der neue Font von TEX aus behandelt? — Der Font bekommt zunächst einmal einen Namen, nämlich `ptmr0` im Falle des EC-codierten *Times-Roman* Fonts; eine `.tfm`-Datei `ptmr0.tfm` beschreibt seine Metrik. Dieser Font könnte jetzt verwendet werden, wenn nicht etliche Zeichen der EC-Codierung (vor allem im Bereich 128–191) nicht im Zeichenvorrat der PostScript-Schriften enthalten wären (siehe Liste am Ende von Anhang B. Diese Zeichen sind zum großen Teil osteuropäische akzentuierte Buchstaben, die in einem virtuellen Font `ptmrq.vf` aus Einzelakzenten und Buchstaben zusammengebaut werden können. Auch die anderen Zeichen werden in dem virtuellen Font auf irgendeine Weise geliefert. TEX erfährt von diesem virtuellen Font nicht mehr als die Metrik der fertigen Zeichen, nämlich `ptmrq.tfm`; vorausgesetzt, die Datei `T1ptm.fd` existiert, die auf `ptmrq` verweist.

Die Datei `ptmrq.vf` verweist wie gewohnt auf den „rohen“, in diesem Fall EC-codierten PostScript-Font, der in der Tabelle `psfonts.map` wie folgt eingetragen ist:

```
ptmr0 Times-Roman " ECEncoding ReEncodeFont " <ec.enc
```

Das bedeutet, daß dvips beim Zugriff auf den PostScript-Font zunächst die Schrift Times-Roman laden muß; diese wird dann mit dem in Anführungszeichen stehenden PostScript-Code umcodiert. Außerdem muß die *ECEncoding* vorab als Header-Datei geladen werden, damit sie dem PostScript-Drucker bekannt ist.

Und wie erzeugt man nun solche umcodierten Fonts? Auf PostScript-Seite wird der rohe Font von dvips bereitgestellt. Seine `.tfm`-Datei erhalten wir mit

```
afm2tfm ptmr.afm -T ec.enc -v ptmrq.vpl ptmr0.tfm
```

Der virtuelle Font für die fehlenden Zeichen wird schließlich durch das Hilfsprogramm vpltovpl von S. Rahtz konstruiert:

```
vpltovpl ptmrq.vpl ptmr.afm
vptovf ptmrq.vpl ptmrq.vf ptmrq.tfm
```

ptmr0 Times-Roman "ECEncoding ReEncodeFont" |<ec.enc

7.3 Nichtresidente PostScript-Fonts

Haben Sie zusätzliche PostScript-Schriften gekauft oder aus der Public Domain besorgt, wie können Sie diese Schriften mit TEX einsetzen? Nach dem oben gesagten sollten es Ihnen klar sein, daß Sie zunächst mal die `.afm`-Datei zu der Schrift brauchen, um daraus alle gewünschten `.tfm`- und `.vf`-Dateien erzeugen zu können. Wie das geht, haben wir ja besprochen.

Wie bringt man dem Drucker dann die neue Schrift bei? dvips folgt den `MAPFONT`-Einträgen in der `.vf`-Datei und erkennt anhand der Liste in der Datei `psfonts.map`, daß es sich um eine PostScript-Schrift handelt. Dort ist für nichtresidente Schriften angegeben, welche Schriftdateien an den Anfang der PostScript-Datei gestellt werden müssen, um alle benötigten Schriften zu definieren (erinnern Sie sich bitte an Abschnitt 3.4). Type-1 Schriften bestehen für gewöhnlich aus einer Datei mit der Namensendung `.pfa` oder `.pfb`, je nachdem, ob die Schrift als lesbares (ASCII-) PostScript-Programm oder dessen binäre Variante vorliegt. dvips akzeptiert beide Versionen und wandelt die binäre Darstellung für den Drucker in die ASCII-Form um.

Beispiel: Die Utopia-Schrift

Damit das folgende Beispiel etwas weniger trocken ist, verwende ich den Public-Domain-Font Utopia als Grundlage (der wie immer mit im Paket enthalten ist). Es gibt vier Varianten von *Utopia*: *Utopia-Regular*, *Utopia-Italic*, *Utopia-Bold* und *Utopia-BoldItalic*. Die Schrift stammt von Adobe und ist freundlicherweise öffentlich freigegeben. Die Type-1-Schriftdateien heißen `putr.pfa`, `putri.pfa`, `putb.pfa` und `putbi.pfa`, die `.afm`-Dateien haben entsprechende Namen.

Als erstes schicken Sie alle `.afm`-Dateien durch afm2tfm und produzieren virtuelle Fonts mit OT1-Codierung:

```
afm2tfm putr.afm -v putr.vpl rputr.tfm
rputr Utopia-Regular
vptovf putr.vpl putr.vf putr.tfm
afm2tfm putri.afm -v putri.vpl rputri.tfm
rputri Utopia-Italic
```

```
vptovf putri.vpl putri.vf putri.tfm
...
```

Dabei weiß afm2tfm nicht, daß diese Schriften keine residenten Druckerschriften sind. Seine Ausgaben zur Eintragung in `psfonts.map` müssen noch um die Angabe der externen Font-Datei erweitert werden:

```
rputr Utopia-Regular <putr.pfa
rputri Utopia-Italic <putri.pfa
...
```

Dadurch kopiert dvips die `.pfa`-Datei wie jede andere Header-Datei mit in die PostScript-Datei hinein, sobald die Schrift in einem Dokument auftaucht. Das setzt voraus, daß Sie die zusätzlichen PostScript-Schriften im Verzeichnis `EMTEX\PS` bzw. `/usr/local/lib/texmf/dvips` unterbringen.

Für TEX müssen Sie noch die Datei `OT1put.fd` schreiben, was Ihnen aber nach der Lektüre des NFSS-Kapitels nicht schwerfallen dürfte. So sieht sie aus:

```
\DeclareFontFamily{OT1}{put}{}{}
\DeclareFontShape{OT1}{put}{m}{n}{
   <-> putr }{}
\DeclareFontShape{OT1}{put}{m}{it}{
   <-> putri }{}
\DeclareFontShape{OT1}{put}{b}{n}{
   <-> putb }{}
\DeclareFontShape{OT1}{put}{b}{it}{
   <-> putbi }{}
```

So, und nun können Sie sich zurücklehnen und mit

```
\fontfamily{put}\selectfont
```

die Ergebnisse dieser Arbeit betrachten. Wenn Sie Lust haben, konnen Sie sich einen ganzen Reigen von Schriftvarianten erschaffen, slanted, SmallCaps, bold extended usw., und sich einen Style `nfutopia` schreiben, mit dem Sie ganze Dokumente in diesen Schriften setzen. *Das ist eine gute Alternative zu den Standard-PostScript-Schriften, die Sie ja nicht kostenlos in angemessener Qualitat bekommen konnen.*

7.4 pk-Dateien für virtuelle Fonts

Es ist möglich, daß Sie zu einer PostScript-Schrift (ob resident oder nicht) `.pk`-Dateien haben möchten.

Stellen Sie sich vor, Sie wollen einen normalen Previewer benutzen, etwa `dviscr` oder `xdvi`. Wenn dieser Previewer mit virtuellen Fonts umgehen kann, wird er wie beschrieben z.B. von dem Schriftnamen `ptmr` zu der „rohen“ PostScript-Schrift `rptmr` finden. Der Previewer interessiert

sich nicht dafür, ob diese Schrift nun druckerresident ist oder nicht. Er braucht in jedem Fall eine Datei `rptmr.pk` (MS-DOS) bzw. `rptmr.300pk` (Unix), um die Schrift auf den Bildschirm zu bringen.

Damit Sie sich Dokumente mit PostScript-Schriften auch ohne GhostScript ansehen können, sind also die passenden `.pk`-Dateien nötig. Die sind zum einen in einem Satz verschiedener Größen auf FTP-Servern erhältlich.

Sollten Sie aber die Original-PostScript-Schriften als Dateien vorliegen haben (z.B. vom Adobe Type Manager), gibt es einen besseren Weg: Mit dem Programm **ps2pk** ist es möglich, aus beliebigen Type-1-Fonts `.pk`-Dateien zu erzeugen. Für Unix ist sogar ein modifiziertes `MakeTeXPK`-Script dabei, das von **xdvi** ab Patchlevel 14 aufgerufen werden kann. Dieses Script entscheidet selbst, ob es sich um einen PostScript-Font handelt, der mit ps2pk „übersetzt“ wird, oder ob wie gewohnt METAFONT gerufen werden muß.

Anhang A

Installation von Karl Berrys TEX-Paket

A.1 Vorhandene TEX-Pakete für Unix

Es gibt zwei Möglichkeiten der TEX-Installation unter Unix; zum einen die Standard-Distribution, die das WEB-System und die WEB-Quellen für TEX, METAFONT „and friends“ enthält, zum anderen eine spezielle Unix-Version, die von dem Unix-Koordinator der TEX Users Group, Karl Berry, bearbeitet und auf dem neuesten Stand gehalten wird. Darin sind nicht nur die WEB-Quellen eingeschlossen, sondern eben auch die dvi-Treiber dvips und xdvi, die sonst separat zu installieren sind. Ich persönlich gebe der Version von Karl Berry den Vorzug, da sie eine zusätzliche, allen Programmen gemeinsame Library enthält: Die `kpathsea`-Library dient zum schnellen Auffinden der benötigten Dateien in den Verzeichnisbäumen, die bei TEX inzwischen monströse Größen erreichen.

Karl Berrys Version besteht aus den Dateien

```
lib-6.1.tar.gz
web-6.1.tar.gz und web2c-6.1.tar.gz
dvipsk-5.55a.tar.gz
xdvik-1.8.tar.gz
```

A.2 Vorbereitung des TEX-Stammverzeichnisses

Zuerst packen Sie die Datei `lib-6.1.tar.gz` im Verzeichnis

```
/usr/local/lib
```

aus. Damit wird ein Startset für TEX angelegt. Das Hauptverzeichnis für alle TEX-bezogenen Dateien ist `/usr/local/lib/texmf`. Die grobe Struktur darunter entnehmen Sie Abbildung A.1.

Es ist wichtig, daß Sie zuerst diese gefüllte Verzeichnisstruktur haben, oder wenigstens die Macro-Dateien (`plain.tex`, `lplain.tex` und zugehörige) sowie die `.tfm`-Dateien (CM-Fonts

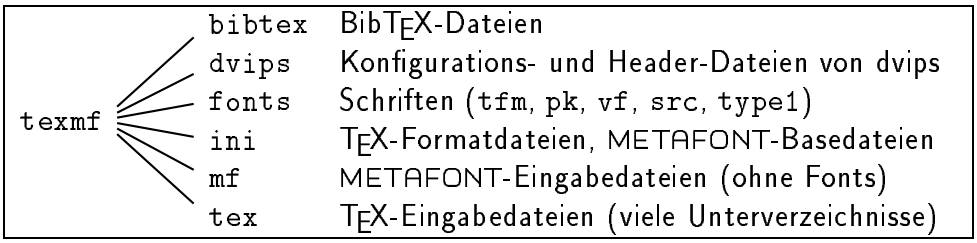


Abbildung A.1: *Verzeichnisstruktur von Karl Berrys Unix-TeX*

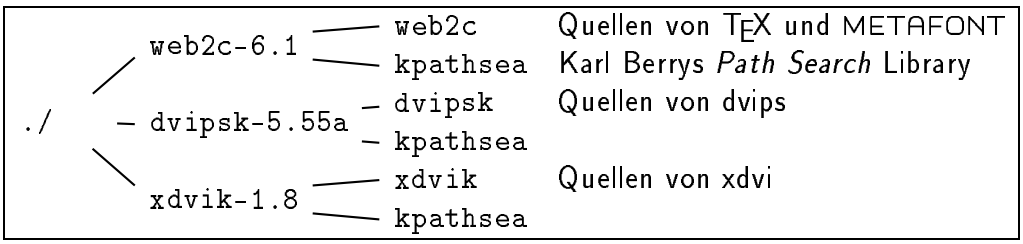


Abbildung A.2: *Verzeichnisstruktur nach dem Auspacken der Quelldateien von Karl Berrys Unix-TeX Distribution.*

und LaTeX-Fonts) an den vergesehenen Platz in der Struktur bringen. Das alles ist nötig, weil im folgenden `make`-Lauf automatisch nach dem Compilieren die Format-Dateien für plainTeX und LaTeX erzeugt werden!

A.3 Die Quelldateien

Die anderen Dateien werden in ein beliebiges Arbeitsverzeichnis ausgepackt, wobei Sie mit etwa 20 MB Platzbedarf rechnen müssen. Beim Auspacken der Dateien entstehen drei Verzeichnisse mit jeweils zwei Unterverzeichnissen (s. Abb. A.2).

Zwar sind die Voreinstellungen der Pfade in allen Paketen gleich, dennoch wäre

es günstig, alle Quellen unter ein Verzeichnis zu bringen, damit alle die gleiche `kpathsea`-Library mit nur einem Satz Einstellungen verwenden. So würden alle Programme gemeinsam und konsistent konfiguriert, was besonders hilfreich wäre, wenn man von den Voreinstellungen abweichen wollte.

A.4 Die Installation

Bei dem `xdvik`-Paket ist neuerdings ein zusätzliches Fenster zur Dateiauswahl (wie bei Ghostview) dabei. Um es zu aktivieren, muß zum einen das Symbol `SELFILE` gesetzt sein, zum anderen müssen Sie in der Datei

```
xdvik/Makefile.in
```

in der folgenden Zeile das Kommentarzeichen entfernen:

```
SELFILE = #Dir.o Draw.o Path.o SelFile.o xgetcwd.o
```

Je nach Entwicklungsstand der Pakete können verschiedene Versionen der `kpathsea`-Library in den drei Verzeichnissen vorliegen. Z.B. braucht dvips in der angegebenen Version eine Datei `kpathsea/tex-file.h`, die in dem älteren `web2c`-Paket noch nicht enthalten ist. Daher ist es momentan anzuraten, lieber in jedem Verzeichnis getrennt zu installieren. Dazu genügen die Befehle

```
sh configure
make "CFLAGS=Spezialeinstellungen"
```

Die Variable `CFLAGS` ist auf '-g' voreingestellt. Für die verschiedenen Programme lesen Sie am besten die `INSTALL`-Dateien in den Quellverzeichnissen. Folgende Einstellungen waren z.B. bei einer Installation unter Linux für einen Tintenstrahldrucker mit 360 dpi sinnvoll:

```
web2c :  CFLAGS=-O -DNO_FOIL_X_WCHAR_T
dvipsk:  CFLAGS=-O -DEMTEX -DTPIC -DDEFRES=360
xdvik :  CFLAGS=-O -DBDPI=360 -DBUTTONS -DSELFILE
                -DA4 -DNO_FOIL_X_WCHAR_T
```

Die eigenartige Definition `NO_FOIL...` ist bei Linux nötig, wenn Definitionen des Typs `wchar_t` in den X- und gcc-Includedateien kollidieren, und muß dann bei allen Programmen vorgenommen werden, die mit X zu tun haben (xdvi und METAFONT für dessen Anzeigemodus `screenstrokes`). Außerdem ist eine weitere Kollision zu erwähnen, die bei der Deklaration der Routine `alloca` auftritt. Die Datei `web2c/web2c/web2c.h` bindet mit der ersten Zeile

```
#include "config.h"
```

indirekt über `stdlib.h` die gcc-Definition von `alloca` ein, definiert aber selbst in der letzten Zeile der Datei

```
extern void *alloca();
```

Diese Zeile muß nur an den Anfang der Datei verschoben werden, damit `alloca` als eigene und nicht als Systemroutine definiert wird.

Bei einer Sun-Installation mit gcc trat das Problem auf, daß in der Datei `xdvik/xgetcwd.c` das Makro `PATH_MAX` nicht definiert war. Hier brauchte nur eine sinnvolle Definition nachgeholt zu werden.

Das waren die einzigen Hindernisse, die mir bei Installationen bisher untergekommen sind. Natürlich können neue Versionen der Pakete andere Probleme mit sich bringen, aber *es ist ausgesprochen erfreulich, daß der gesamte komplexe Ablauf auf IBM- und HP-Workstations mit gcc sowie SiliconGraphics mit cc völlig reibungslos abläuft*, und mit den kleinen Änderungen auch die Linux- und Sun-Installation klappen.

Nachdem make erfolgreich durchgelaufen ist, können Sie die Programme, Manual Pages und weitere Daten mit

```
make install
```

in jedem der drei Verzeichnisse installieren. Dazu brauchen Sie die Schreibrechte in den Zielverzeichnissen

```
/usr/local/bin,
/usr/local/man,
/usr/local/info und
/usr/local/lib/texmf
```

Damit haben Sie ein funktionsfähiges TEX mit zwei dvi-Treibern. Allerdings enthält es nur die englischen Trenntabellen. Sie werden sich daher noch die Dateien

```
ghyph31.tex und german3.sty
```

besorgen müssen und neue Formate erstellen.

A.5 Zur kpathsea-Library

Zweck dieser Library ist es, gesuchte Dateien so schnell wie möglich in der oft immens großen Unterverzeichnisstruktur von TEX zu finden. Gleichzeitig wird ein Alias-Mechanismus zur Verfügung gestellt, um Probleme mit langen Dateinamen etwa unter MS-DOS aus dem Weg zu gehen. Dazu sind zwei Dateien erforderlich.

Die Datei `/usr/local/lib/texmf/ls-R` enthält die Liste aller Dateien im TEX-Stammverzeichnis. Sie wird erzeugt bzw. auf den neuesten Stand gebracht durch

```
cd /usr/local/lib/texmf
ls -R /usr/local/lib/texmf > ls-R ,
```

vorausgesetzt, das Ausgabeformat von `ls -R` stimmt mit dem erwarteten Format überein (was aber fast immer der Fall ist).

Wenn diese Liste veraltet sein sollte, gibt es zwei Möglichkeiten: Entweder gibt es eine in der Liste aufgeführte Datei nicht mehr, dann versagt der Zugriff darauf; oder die gesuchte Datei ist noch nicht in der Liste enthalten, da sie zu neu ist. Falls `kpathsea` eine Datei nicht in der Liste findet, sucht es den Unterverzeichnisbaum ab, also wird die Datei gefunden, wenn auch nicht so schnell wie wenn sie in der Liste stehen würde. Das regelmäßige Erstellen der Liste läßt sich mit `cron` automatisieren.

Die zweite Datei enthält die Alias-Einträge. Sie findet nur für Schriftdateien Verwendung und lautet mit vollem Pfad

```
/usr/local/lib/texmf/fonts/texfonts.map
```

Eine typische Anwendung ist die Austauschbarkeit der verschiedenen Varianten der Schrift `lcircle`, die so aussieht:

```
% These fonts have been named both 'circle10' and 'lcircle10'.
% But 'lcirc10' is best, anyway, as it's short enough for DOS.
% Allow any of these to mean any of the others.
% --karl@cs.umb.edu, 23 Jan 93.
circle10 lcircle10
circle10 lcirc10
lcircle10 circle10
lcircle10 lcirc10
lcirc10 circle10
lcirc10 lcircle10

circlew10 lcirclew10
circlew10 lcircw10
lcirclew10 circlew10
lcirclew10 lcircw10
lcircw10 circlew10
lcircw10 lcirclew10
```

Allgemein haben Einträge folgendes Format: Das erste Wort einer Zeile gibt den echten Dateinamen an, das zweite den Alias. Leerzeilen sowie Zeilen, die mit '%' beginnen, werden ignoriert. Wenn ein Wort eine Namenserweiterung hat, gilt der Alias nur für die Erweiterung. Fehlt die Angabe der Erweiterung, gilt der Alias für alle Namenserweiterungen.

Bei der Suche nach einer Datei wird im Zweifelsfall die real existierende Datei genommen, nicht der Alias.

Anhang B

Codetabellen

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0*x*	Γ	Δ	Θ	Λ	Ξ	Π	Σ	Υ	Φ	Ψ	Ω	ff	fi	fl	ffi	ffl
1*x*	ı	ȷ	`	´	ˇ	˘	¯	°	¸	ß	æ	œ	ø	Æ	Œ	Ø
2*x*		!	”	#	$	%	&	’	(	)	*	+	,	-	.	/
3*x*	0	1	2	3	4	5	6	7	8	9	:	;	¡	=	¿	?
4*x*	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
5*x*	P	Q	R	S	T	U	V	W	X	Y	Z	[	“	]	^	˙
6*x*	‘	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
7*x*	p	q	r	s	t	u	v	w	x	y	z	–	—	”	~	¨

Tabelle B.1: *Codetabelle für die alte TEX-Codierung (OT1) am Beispiel cmr10*

A	A	101	101	Ø	Oslash	351	330	|	bar	174	174
Æ	AE	341	306	Õ	Otilde	—	325	{	braceleft	173	173
Á	Aacute	—	301	P	P	120	120	}	braceright	175	175
Â	Acircumflex	—	302	Q	Q	121	121	[	bracketleft	133	133
Ä	Adieresis	—	304	R	R	122	122	⌋	bracketright	135	135
À	Agrave	—	300	S	S	123	123	˘	breve	306	226
Å	Aring	—	305	Š	Scaron	—	—	¦	brokenbar	—	246
Ã	Atilde	—	303	T	T	124	124	•	bullet	267	—
B	B	102	102	Þ	Thorn	—	336	c	c	143	143
C	C	103	103	U	U	125	125	ˇ	caron	317	237
Ç	Ccedilla	—	307	Ú	Uacute	—	332	ç	ccedilla	—	347
D	D	104	104	Û	Ucircumflex	—	333	¸	cedilla	313	270
E	E	105	105	Ü	Udieresis	—	334	¢	cent	242	242
É	Eacute	—	311	Ù	Ugrave	—	331	ˆ	circumflex	303	223
Ê	Ecircumflex	—	312	V	V	126	126	:	colon	072	072
Ë	Edieresis	—	313	W	W	127	127	,	comma	054	054
È	Egrave	—	310	X	X	130	130	©	copyright	—	251
Ð	Eth	—	320	Y	Y	131	131	¤	currency	250	244
F	F	106	106	Ý	Yacute	—	335	d	d	144	144
G	G	107	107	Ÿ	Ydieresis	—	—	†	dagger	262	—
H	H	110	110	Z	Z	132	132	‡	daggerdbl	263	—
I	I	111	111	Ž	Zcaron	—	—	°	degree	—	260
Í	Iacute	—	315	a	a	141	141	¨	dieresis	310	250
Î	Icircumflex	—	316	á	aacute	—	341	÷	divide	—	367
Ï	Idieresis	—	317	â	acircumflex	—	342	$	dollar	044	044
Ì	Igrave	—	314	´	acute	302	222	˙	dotaccent	307	227
J	J	112	112	´	acute	302	264	1	dotlessi	365	220
K	K	113	113	ä	adieresis	—	344	e	e	145	145
L	L	114	114	æ	ae	361	346	é	eacute	—	351
Ł	Lslash	350	—	à	agrave	—	340	ê	ecircumflex	—	352
M	M	115	115	&	ampersand	046	046	ë	edieresis	—	353
N	N	116	116	å	aring	—	345	è	egrave	—	350
Ñ	Ntilde	—	321	^	asciicircum	136	136	8	eight	070	070
O	O	117	117	~	asciitilde	176	176	...	ellipsis	274	—
Œ	OE	352	—	*	asterisk	052	052	—	emdash	320	—
Ó	Oacute	—	323	@	at	100	100	–	endash	261	—
Ô	Ocircumflex	—	324	ã	atilde	—	343	=	equal	075	075
Ö	Odieresis	—	326	b	b	142	142	ð	eth	—	360
Ò	Ograve	—	322	\	backslash	134	134	!	exclam	041	041

¡	exclamdown	241	241	o	o	157	157	s	s	163	163
f	f	146	146	ó	oacute	—	363	š	scaron	—	—
fi	fi	256	—	ô	ocircumflex	—	364	§	section	247	247
5	five	065	065	ö	odieresis	—	366	;	semicolon	073	073
fl	fl	257	—	ú	oe	372	—	7	seven	067	067
f	florin	246	—	˛	ogonek	316	236	6	six	066	066
4	four	064	064	ò	ograve	—	362	/	slash	057	057
⁄	fraction	244	—	1	one	061	061		space	040	040
g	g	147	147	½	onehalf	—	275	£	sterling	243	243
ß	germandbls	373	337	¼	onequarter	—	274	t	t	164	164
`	grave	301	221	¹	onesuperior	—	271	þ	thorn	—	376
>	greater	076	076	ª	ordfeminine	343	252	3	three	063	063
«	guillemotleft	253	253	º	ordmasculine	353	272	¾	threequarters	—	276
»	guillemotright	273	273	ø	oslash	371	370	³	threesuperior	—	263
‹	guilsinglleft	254	—	õ	otilde	—	365	~	tilde	304	224
›	guilsinglright	255	—	p	p	160	160	™	trademark	—	—
h	h	150	150	¶	paragraph	266	266	2	two	062	062
˝	hungarumlaut	315	235	(	parenleft	050	050	²	twosuperior	—	262
-	hyphen	055	255	)	parenright	051	051	u	u	165	165
i	i	151	151	%	percent	045	045	ú	uacute	—	372
í	iacute	—	355	.	period	056	056	û	ucircumflex	—	373
î	icircumflex	—	356	·	periodcentered	264	267	ü	udieresis	—	374
ï	idieresis	—	357	‰	perthousand	275	—	ù	ugrave	—	371
ì	igrave	—	354	+	plus	053	053	_	underscore	137	137
j	j	152	152	±	plusminus	—	261	v	v	166	166
k	k	153	153	q	q	161	161	w	w	167	167
l	l	154	154	?	question	077	077	x	x	170	170
<	less	074	074	¿	questiondown	277	277	y	y	171	171
¬	logicalnot	—	254	"	quotedbl	042	042	ý	yacute	—	375
ł	lslash	370	—	„	quotedblbase	271	—	ÿ	ydieresis	—	377
m	m	155	155	“	quotedblleft	252	—	¥	yen	245	245
¯	macron	305	257	”	quotedblright	272	—	z	z	172	172
-	minus	—	055	‘	quoteleft	140	140	ž	zcaron	—	—
µ	mu	—	265	’	quoteright	047	047	0	zero	060	060
×	multiply	—	327	,	quotesinglbase	270	—				
n	n	156	156	'	quotesingle	251	—				
9	nine	071	071	r	r	162	162				
ñ	ntilde	—	361	®	registered	—	256				
#	numbersign	043	043	˚	ring	312	232				

Zeichen der ECEncoding, die nicht im Standard-PostScript-Zeichensatz definiert sind und daher in einem virtuellen Font zusammengebaut oder sonstwie erzeugt werden müssen:

	compoundwordmark	027	IJ	IJ	234
ȷ	dotlessj	032	İ	Idotaccent	235
ff	ff	033	đ	dbar	236
ffi	ffi	036	ă	abreve	240
ffl	ffl	037	ą	aogonek	241
Ă	Abreve	200	ć	cacute	242
Ą	Aogonek	201	č	ccaron	243
Ć	Cacute	202	ď	dquoteright	244
Č	Ccaron	203	ě	ecaron	245
Ď	Dcaron	204	ę	eogonek	246
Ě	Ecaron	205	ğ	gbreve	247
Ę	Eogonek	206	Í	lacute	250
Ğ	Gbreve	207	ľ	lquoteright	251
Ĺ	Lacute	210	ń	nacute	253
Ľ	Lquoteright	211	ň	ncaron	254
Ń	Nacute	213	ŋ	eng	255
Ň	Ncaron	214	ő	ohungarumlaut	256
Ŋ	Eng	215	ŕ	racute	257
Ő	Ohungarumlaut	216	ř	rcaron	260
Ŕ	Racute	217	ś	sacute	261
Ř	Rcaron	220	ş	scedilla	263
Ś	Sacute	221	ť	tquoteright	264
Ş	Scedilla	223	ţ	tcedilla	265
Ť	Tcaron	224	ű	uhungarumlaut	266
Ţ	Tcedilla	225	ů	uring	267
Ű	Uhungarumlaut	226	ź	zacute	271
Ů	Uring	227	ż	zdotaccent	273
Ź	Zacute	231	ij	ij	274
Ż	Zdotaccent	233	SS	Germandbls	337

Die einzigen Zeichen, die im Bereich 80–BF vorhanden sind, sind Lslash, Scaron, Ydieresis, Zcaron, section; lslash, scaron, ydieresis, zcaron, exclamdown, questiondown, sterling.

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0x	`	´	ˆ	˜	¨	˝	°	ˇ	˘	¯	˙	¸	¿	¡	‹	›
1x	“	”	„	«	»	–	—		ı	ȷ	ff	fi	fl	ffi	ffl	
2x		!	"	#	$	%	&	’	(	)	*	+	,	-	.	/
3x	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
4x	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
5x	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
6x	‘	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
7x	p	q	r	s	t	u	v	w	x	y	z	{	|	}	~	-
8x	Ă	Ą	Ć	Č	Ď	Ě	Ę	Ğ	Ĺ	Ľ	Ł	Ń	Ň	Ŋ	Ő	Ŕ
9x	Ř	Ś	Š	Ş	Ť	Ţ	Ű	Ů	Ÿ	Ź	Ž	Ż	Ĳ	İ	đ	§
Ax	ă	ą	ć	č	ď	ě	ę	ğ	ĺ	ľ	ł	ń	ň	ŋ	ő	ŕ
Bx	ř	ś	š	ş	ť	ţ	ű	ů	ÿ	ź	ž	ż	ĳ	¡	¿	£
Cx	À	Á	Â	Ã	Ä	Å	Æ	Ç	È	É	Ê	Ë	Ì	Í	Î	Ï
Dx	Đ	Ñ	Ò	Ó	Ô	Õ	Ö	Œ	Ø	Ù	Ú	Û	Ü	Ý	Þ	SS
Ex	à	á	â	ã	ä	å	æ	ç	è	é	ê	ë	ì	í	î	ï
Fx	ð	ñ	ò	ó	ô	õ	ö	œ	ø	ù	ú	û	ü	ý	þ	ß

Tabelle B.2: *Codetabelle für die EC Codierung (T1). Die Akzentzeichen sind in den ersten zwei Zeilen untergebracht. Dann folgt im Gegensatz zur OT1-Codierung der übliche ASCII-Zeichensatz von 32–126. Die nächsten zwei Paare von Zeilen enthalten hauptsächlich die osteuropäischen Sonderzeichen, zuerst als Groß-, dann als Kleinbuchstaben. Die letzten zwei Zeilenpaare entsprechen der ISO Latin-1 Codierung bis auf das kleine und große ‘ß’. PostScript-Schriften lassen sich nicht ohne weiteres zu T1 umcodieren, da etliche Zeichen fehlen. Zwar können die osteuropäischen Akzentkombinationen zum großen Teil durch virtuelle Fonts zusammengebaut werden, aber die eigenständigen Zeichen namens ‘compoundwordmark’ (#23 dez.), ‘dotlessj’ (#26), ‘Eng’ (#141), ‘eng’ (#173) bereiten Schwierigkeiten.*

	x0	x1	x2	x3	x4	x5	x6	x7	x8	x9	xA	xB	xC	xD	xE	xF	
0x																	0x
1x																	1x
2x		!	"	#	$	%	&	’	(	)	*	+	,	-	.	/	2x
3x	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?	3x
4x	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	4x
5x	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_	5x
6x	‘	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	6x
7x	p	q	r	s	t	u	v	w	x	y	z	{	|	}	~		7x
8x																	8x
9x																	9x
Ax		¡	¢	£	⁄	¥	ƒ	§	¤	'	“	«	‹	›	fi	fl	Ax
Bx		–	†	‡	·		¶	•	‚	„	”	»	…	‰		¿	Bx
Cx		`	´	ˆ	˜	¯	˘	˙	¨		°	¸		˝	˛	ˇ	Cx
Dx	—																Dx
Ex		Æ		ª					Ł	Ø	Œ	º					Ex
Fx		æ				ı			ł	ø	œ	ß					Fx
	x0	x1	x2	x3	x4	x5	x6	x7	x8	x9	xA	xB	xC	xD	xE	xF	

Tabelle B.3: *Die Adobe StandardEncoding am Beispiel der Schrift Helvetica*

	*x*0	*x*1	*x*2	*x*3	*x*4	*x*5	*x*6	*x*7	*x*8	*x*9	*x*A	*x*B	*x*C	*x*D	*x*E	*x*F	
0*x*																	**0***x*
1*x*																	**1***x*
2*x*		!	"	#	$	%	&	’	(	)	*	+	,	–	.	/	**2***x*
3*x*	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?	**3***x*
4*x*	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	**4***x*
5*x*	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_	**5***x*
6*x*	‘	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	**6***x*
7*x*	p	q	r	s	t	u	v	w	x	y	z	{	|	}	~		**7***x*
8*x*																	**8***x*
9*x*	ı	`	´	ˆ	˜	¯	˘	˙	¨		˚	¸		˝	˛	ˇ	**9***x*
A*x*		¡	¢	£	¤	¥	¦	§	¨	©	ª	«	¬	-	®	¯	**A***x*
B*x*	°	±	2	3	´	µ	¶	·	¸	1	º	»	¼	½	¾	¿	**B***x*
C*x*	À	Á	Â	Ã	Ä	Å	Æ	Ç	È	É	Ê	Ë	Ì	Í	Î	Ï	**C***x*
D*x*	Đ	Ñ	Ò	Ó	Ô	Õ	Ö	×	Ø	Ù	Ú	Û	Ü	Ý	Þ	ß	**D***x*
E*x*	à	á	â	ã	ä	å	æ	ç	è	é	ê	ë	ì	í	î	ï	**E***x*
F*x*	ð	ñ	ò	ó	ô	õ	ö	÷	ø	ù	ú	û	ü	ý	þ	ÿ	**F***x*
	*x*0	*x*1	*x*2	*x*3	*x*4	*x*5	*x*6	*x*7	*x*8	*x*9	*x*A	*x*B	*x*C	*x*D	*x*E	*x*F	

Tabelle B.4: *Codetabelle nach dem ISO-Standard 8859-1 (Latin-1) am Beispiel der Schrift Helvetica*

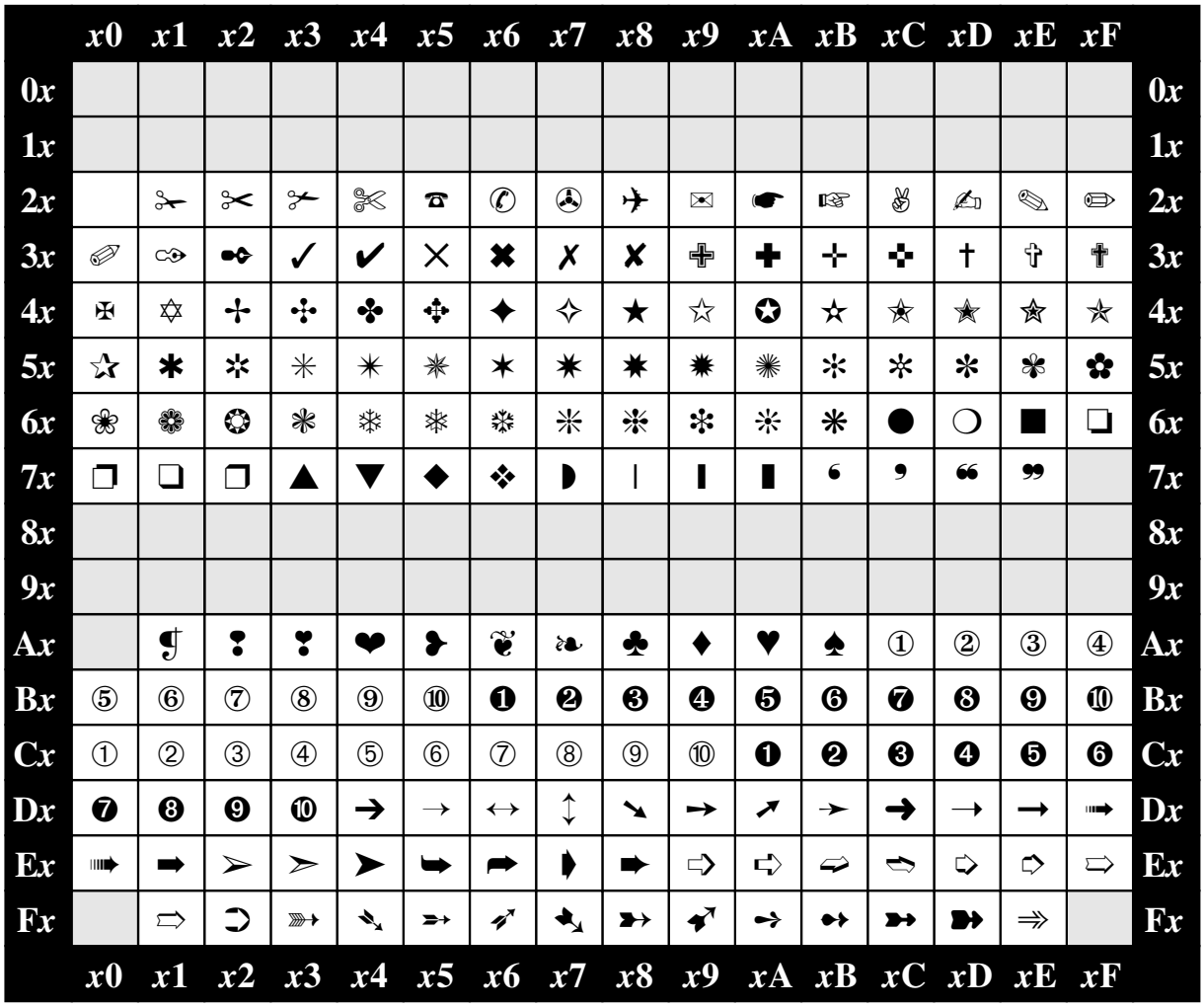


Tabelle B.5: *Codetabelle der PostScript-Schrift ZapfDingbats*

	*x*0	*x*1	*x*2	*x*3	*x*4	*x*5	*x*6	*x*7	*x*8	*x*9	*x*A	*x*B	*x*C	*x*D	*x*E	*x*F	
0*x*																	**0*x***
1*x*																	**1*x***
2*x*		!	∀	#	∃	%	&	∍	(	)	*	+	,	−	.	/	**2*x***
3*x*	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?	**3*x***
4*x*	≅	Α	Β	Χ	Δ	Ε	Φ	Γ	Η	Ι	ϑ	Κ	Λ	Μ	Ν	Ο	**4*x***
5*x*	Π	Θ	Ρ	Σ	Τ	Υ	ς	Ω	Ξ	Ψ	Ζ	[	∴	]	⊥	_	**5*x***
6*x*	‾	α	β	χ	δ	ε	φ	γ	η	ι	φ	κ	λ	μ	ν	ο	**6*x***
7*x*	π	θ	ρ	σ	τ	υ	ϖ	ω	ξ	ψ	ζ	{	|	}	~		**7*x***
8*x*																	**8*x***
9*x*																	**9*x***
A*x*	€	ϒ	′	≤	⁄	∞	*ƒ*	♣	♦	♥	♠	↔	←	↑	→	↓	**A*x***
B*x*	°	±	″	≥	×	∝	∂	•	÷	≠	≡	≈	…	|	—	↵	**B*x***
C*x*	ℵ	ℑ	ℜ	℘	⊗	⊕	∅	∩	∪	⊃	⊇	⊄	⊂	⊆	∈	∉	**C*x***
D*x*	∠	∇	®	©	™	∏	√	·	¬	∧	∨	⇔	⇐	⇑	⇒	⇓	**D*x***
E*x*	◊	⟨	®	©	™	∑	⎛	⎜	⎝	⎡	⎢	⎣	⎧	⎨	⎩	⎪	**E*x***
F*x*		⟩	∫	⌠	⎮	⌡	⎞	⎟	⎠	⎤	⎥	⎦	⎫	⎬	⎭		**F*x***
	*x*0	*x*1	*x*2	*x*3	*x*4	*x*5	*x*6	*x*7	*x*8	*x*9	*x*A	*x*B	*x*C	*x*D	*x*E	*x*F	

Tabelle B.6: *Codetabelle der PostScript-Schrift Symbol*

Anhang C

Nomenklatur der PostScript-Schriften

Zu jeder PostScript-Schrift gibt es `.tfm`- und `.vf`-Dateien. Wegen der Einschränkung auf acht Zeichen lange Dateinamen in MS-DOS muß es eine Kurzform der Namen geben, die aber trotzdem eindeutig einer Schrift zugeordnet werden kann. Dafür hat Karl Berry ein Kürzelschema entwickelt, das den Kürzeln bei der Schriftenauswahl des NFSS verwandt ist. Zur Katalogisierung der Schriften werden ebenfalls die Attribute Familie, Gewicht, Form und Breite verwendet. Es gibt darüber hinaus verschiedene Versionen einer Schriftfamilie von mehreren Herstellern. Daher muß auch der Hersteller einbezogen werden. Ganz detailliert finden Sie das Schema in der Original-Anleitung zu dvips oder in der aktuellen Version der Datei

`fontname-1.5.tar.gz` auf dem Server `ftp.cs.umb.edu`

Der Acht-Zeichen-Code

Die Aufteilung der acht Zeichen ist Q-SS-G-F-B-DD. Dabei ist

- Q die Quelle der Schrift (etwa GNU oder Adobe)
- SS die Schriftfamilie
- G ihr Gewicht (*weight*)
- F die Form (*shape*)
- B die Breite (*expansion*)
- DD die Design-Größe der Schrift.

Die Design-Größe entfällt, wenn die Schrift skalierbar ist (wie bei allen Standard-PostScript-Schriften).
Wenn die Breite „normal“ ist, wird sie weggelassen.

Q	Quelle der Schrift		
9	unbekannte Quelle	m	Monotype
a	Autologic	n	IBM
b	Bitstream	p	Adobe PostScript
c	Agfa-Compugraphic	s	Sun
d	Digital Typeface Corporation	u	URW
f	frei verfügbar (PD)	x	American Mathematical Society
g	Free Software Foundation (GNU)	y	Y&Y
h	Bigelow & Holmes		
i	International Typeface Corporation	r	Präfix für virtuelle Fonts
l	Linotype	z	Präfix für „bizarre“ Fonts

Tabelle C.1: *Quellenkürzel in Karl Berrys Schema. Die Kürzel 'r' und 'z' erscheinen zusätzlich zu wirklichen Quellen. „Bizarr“ meint solche Fonts, die nicht nach den gängigen Merkmalen klassifizierbar sind.*

Wenn Breite und Gewicht „normal“ sind, werden beide weggelassen.
Ausnahmen bestehen für virtuelle Fonts, dazu weiter unten.

Die folgenden Listen (Tabellen C.1–C.5) zeigen den aktuellen Stand der Kürzel:

Verschiedene Codierungen sind mit im Formkürzel untergebracht. Das betrifft vor allem die virtuellen Fonts: Ganz nach Belieben kann jedermann neue virtuelle Fonts aus anderen Schriften zusammenbauen, alles ist erlaubt. Abgesehen von diesen indivuellen Schöpfungen, die sich diesem Namensschema entziehen, gibt es vier Standardfälle, bei denen eine Systematik möglich ist:

1. PostScript-Schrift, OT1-codiert. Diese Schrift erhält ihr Kürzel nur aufgrund der Schriftattribute, z.B.

   ```
   pagk   AvantGarde-Book
   pbkd   Bookman-Demi
   pncri  NewCenturySchlbk-Italic
   pplbu  Palatino-BoldUnslanted
   ```

2. PostScript-Schrift in *AdobeStandardEncoding*. Diese Schrift bekommt ein 'r' vor den Namen gesetzt: `rpagk` ...

3. PostScript-Schrift in *ECEncoding* (soweit die Zeichen in der PostScript-Schrift schon definiert sind). Diese Schrift bekommt eine '0' angehängt: `pagk0` ...

4. PostScript-Schrift, T1-codiert. Diese Schrift bekommt ein 'q' als zweites Formkürzel angehängt: `pagkq` ...

SS	Schriftfamilie		
ad	Adobe Garamond	lg	Letter Gothic
ag	*Avant Garde*	nc	*New Century Schoolbook*
bk	*Bookman*	nt	Times New Roman
bu	Brush	oe	Old English
bv	Baskerville	op	Optima
ca	Caslon	pl	*Palatino*
ch	Charter	sc	Script
cm	Computer Modern	st	Stone
cr	*Courier*	sy	*Symbol*
eu	Euler	tm	*Times*
fu	Futura	un	Univers
gm	Garamond	ut	Utopia
hv	*Helvetica*	zc	*Zapf Chancery*
lc	Lucida	zd	*Zapf Dingbats*

Tabelle C.2: *Schriftfamilienkürzel in Karl Berrys Schema. Standardschriften sind hervorgehoben. Die dreistellige Kombination aus Quelle und Schriftfamilie entspricht dem Familienkürzel der PostScript-Schriften in NFSS. Manche Schriften haben je nach Anbieter unterschiedliche Namen, dann gelten Konventionen. Zum Beispiel sind Arial und Times New Roman von Monotype Nachahmungen der Helvetica und Times von Linotype, wobei letztere zu Adobe gerechnet werden, weil sie bei jedem PostScript-Drucker dabei sind.*

G	Gewicht				
a	hairline	r	regular	x	extra bold
t	thin	m	medium	h	heavy
i	extra light	d	demi	c	black
l	light	s	semi	u	ultra
k	book	b	bold	p	poster

Tabelle C.3: *Gewichtskürzel in Karl Berrys Schema (sortiert).*

F	Form		
0	Adobe standard enc.	4	fax
1	semi sans	5	Lautschrift
2	nur veränderte Zeichen	6	semi serif
3	Brüche	9	Oldstyle-Ziffern
a	Adobe alternate enc.	n	informal
b	bright	o	oblique (slanted)
c	small caps	p	ornament
d	display	q	T1/Cork encoding
e	engraved	r	normal (roman or sans)
f	Fraktur	s	sans serif
g	grooved (IBM Logo)	t	typewriter
h	shadow	u	unslanted italic
i	italic	v	math extension
j	invisible	w	script, Handschrift
k	Greek	x	Adobe expert enc.
l	outline	y	symbol
m	math italic	z	cyrillic

Tabelle C.4: *Formkürzel in Karl Berrys Schema. Wenn eine Schrift wie Helvetica keine Serifen hat, gilt das als „normal", nicht als „sans serif". Ebenso für „typewriter". Manche Schriften haben mehrere Formangaben, z.B. Stone Informal Italic. In so einem Fall muß man wohl oder übel mehrere Formkürzel schreiben.*

B	Breite		
u	ultra compressed	t	thin
o	ultra condensed	r	regular, normal,
q	extra compressed,		medium
	extra condensed	x	extended (von Hand)
c	condensed (von Hand)	e	expanded (automatisch)
n	narrow (automatisch)	w	wide

Tabelle C.5: *Breitenkürzel in Karl Berrys Schema.*

Anhang D

Literaturverzeichnis

PostScript

Bücher...

Adobe Systems Incorporated: Adobe Type 1 Font Format, Addison-Wesley, Reading, Mass., 1990.

Adobe Systems Incorporated: PostScript Language Reference Manual, Second Edition, Addison-Wesley, Reading, Mass., 1991.

Thomas Merz: TerminalBuch PostScript — Fonts und Programmiertechnik, Oldenbourg Verlag, München, 1991.

Dušan Živadinović: Bücher zum Thema PostScript (Buchkritiken), c't 5/92, S. 258 ff.

Zeitschriftenartikel...

Martin Cracauer: Geisterstunde — GNU-GhostScript 2.3: PD-Interpreter für PostScript, iX 3/92, S. 40 ff.

Rainer Klute: Zweite Geisterstunde — Neue GhostScript-Version 2.4 und Ghostview, iX 4/92, S. 68 f.

Wilfried Söker: Aufstiegsberatung — Erste Erfahrungen mit Adobes neuer Seitenbeschreibungssprache, iX 2/93, S. 106 ff.

Grafik

Bücher...

Foley, van Dam, Feiner, Hughes: Computer Graphics, Principles and Practice, Second Edition, Addison-Wesley, Reading, Mass., 1990.

Friedhelm Sowa: Grafikintegration in TEX, in: *DANTE e.V. (Hrsg.)*, Offizin, Addison-Wesley, Bonn, 1994.

Zeitschriftenartikel...

Heinz Werntges: Von Rahmen und Bildern — Grafikimport in TeX/LaTeX, c't 12/92, S. 252 ff.

Thomas Merz: Stufe zwei — Erweiterung im großen Stil, PostScript Level 2, c't 8/93, S. 182 ff.

Oliver Kesy: Bilder schrumpfen — Neue Methoden und Programme zur Bildkompression, c't 11/93, S. 120 ff.

Thomas Merz: Blick in die Kapsel — Das Grafikformat Encapsulated PostScript, c't 12/93, S. 240 ff.

Ulrich Allgeier: Formatkünstler. Pizazz Plus, SnapPRO! und HiJaak PRO — Grafik-Utilities für Windows, c't 4/94, S. 192 ff.

Thomas Merz: Gut verpackt — Drucken von JPEG-Bildern mit PostScript Level 2, c't 6/94, S. 236 ff.

Schriften und Typografie

Bücher...

Hans Ed. Meier: Schriftgestaltung mit Hilfe des Computers. Typographische Grundregeln, ETH Zürich, Departement Informatik, Institut für Computersysteme, 1990.

Ulrich Hilgefort: Bücher zum Thema Typografie (Buchkritiken), c't 4/93, S. 282 ff.

Zeitschriftenartikel...

Carsten Meyer, Udo Flohr: True Stories — Was ist dran am TrueType-Font?, c't 7/91, S. 144 ff.

Bernd Behr: Welt der Zeichen — Neuer Zeichensatzstandard der ISO, c't 9/92, S. 241 ff.

Klaus-Peter Karmann, Ulrich Hilgefort: Schriften-Krämer — Softfonts bearbeiten und über Systemgrenzen konvertieren, c't 4/93, S. 164 ff.

Ulrich Hilgefort: Typen mit Charakter — Systematik der Schriften und ihr Familienleben, c't 4/93, S. 140 ff.

Thomas Merz: Zeichen gesetzt — PostScript, TrueType und der ganze Rest, c't 4/93, S. 150 ff.

Lucas de Groot, Thomas Merz: Schriftenwerk(l)er — Fonteditoren und -konverter für Windows, c't 1/94, S. 88 ff.

Sonstiges

Zeitschriftenartikel

Stefan Mintert: LaTeX lebt — Was bringt LaTeX 2_{ε}? iX 6/94, S. 184 ff.

Programmanleitungen, Dokumentation

Tomas Rokicki: DVIPS: A TEX Driver, 1993 (Version 5.518)

Timothy van Zandt: PSTricks: PostScript macros for Generic TEX. User's Guide, März 1993 (Version 0.93)

Frank Mittelbach, Rainer Schöpf: A New Font Selection Scheme for TEX Macro Packages, Tugboat 10#2, 1989, S. 222–238.

Schriften des HRZ Gießen

Stefan Hofstetter: „TEX und PostScript — ein Traumpaar, oder: Eine Anleitung für das Programm DVIPS und die zugehörigen Hilfsprogramme“, HRZ Uni Gießen, 1992.

Johannes Hoffstadt: „Installation und Anwendung von TEX auf Unix-Workstations“, HRZ Uni Gießen, 1993.

Ausgewählte Literatur zu TEX, LATEX und METAFONT

Donald E. Knuth: Computers&Typesetting, Vol. A: The TEXbook, Addison-Wesley, Reading, Mass., 1991.

Donald E. Knuth: Computers&Typesetting, Vol. C: The METAFONTbook, Addison-Wesley, Reading, Mass., 1986.

Leslie Lamport: LATEX, A Document Preparation System, Addison-Wesley, Reading, Mass., 1986.

Raymond Seroul, Silvio Levy: A Beginner's Book of TEX, Springer-Verlag, New York, 1991.

Wynter Snow: TEX for the Beginner (with LATEX notes), Addison-Wesley, Reading, Mass., 1992.

Reinhard Wonneberger: Kompaktführer LaTeX, 3., erw. Auflage, Addison-Wesley, Bonn, 1993.

Michel Goossens, Frank Mittelbach, Alexander Samarin: The LaTeX Companion, Addison-Wesley, Reading, Mass., 1994.