Die TEXnische Komödie

DANTE
Deutschsprachige
Anwendervereinigung TEX e.V.

11. Jahrgang Heft 1/1999 Februar 1999

1/1999

Impressum

„Die TEXnische Komödie“ ist die Mitgliedszeitschrift von DANTE e.V. Der Bezugspreis ist im Mitgliedsbeitrag enthalten. Namentlich gekennzeichnete Beiträge geben die Meinung der Schreibenden wieder. Reproduktion oder Nutzung der erschienenen Beiträge durch konventionelle, elektronische oder beliebige andere Verfahren ist nur im nicht-kommerziellen Rahmen gestattet. Verwendungen in größerem Umfang bitte zur Information bei DANTE e.V. melden.

Beiträge sollten in Standard-LATEX-Quellcode unter Verwendung der Dokumentenklasse `dtk` erstellt und an untenstehende Anschrift geschickt werden (entweder per E-Mail oder auf Diskette). Sind spezielle Makros, LATEX-Pakete oder Schriften dafür nötig, so müssen auch diese mitgeliefert werden. Außerdem müssen sie auf Anfrage Interessierten zugänglich gemacht werden.

Diese Ausgabe wurde mit Hilfe folgender Programme fertiggestellt: `TeX, Version 3.14159 (Web2c 7.2)`, `LaTeX2e <1998/06/01>`, `xdvik 18f` und `windvi` (für die Bildschirmdarstellung) und `dvips(k) 5.78` (für Korrektur und Belichtung). Die Schriften zur Belichtung wurden mit dem METAFONT-Mode `linoone` (1270 dpi) berechnet.

Erscheinungsweise: vierteljährlich
Erscheinungsort: Heidelberg
Auflage: 2300

Herausgeber: DANTE, Deutschsprachige Anwendervereinigung TEX e.V.
Postfach 10 18 40
69008 Heidelberg

E-Mail: `dante@dante.de`
`dtk-redaktion@dante.de` (Redaktion)

Druck: Konrad Triltsch Druck- und Verlagsanstalt GmbH
Haugerring 5
97070 Würzburg

Redaktionsschluß für Heft 2/1999: 19. März 1999 ISSN 1434-5897

Editorial

Liebe Leserinnen und Leser,

eine neues Jahr bringt auch für „Die TEXnische Komödie“ etwas Neues. Nein, nicht die neue Rechtschreibung. Die hatten wir zwar schon in einem einzelnen Beitrag – wer hatte das bemerkt? Aber generell wollen wir diese noch nicht durchgängig umsetzen, auch wenn diese Änderung irgendwann ansteht.

Ab dieser Ausgabe benutzen wir standardmäßig die EC-Schriften. Diese sind zwar nicht mehr brandheiß. Aber „Die TEXnische Komödie“ hatte sich schon immer eine gewisse Zurückhaltung bei Neuerungen auf die Fahnen geschrieben. Im Zusammenhang mit den EC-Schriften gab es immer wieder Diskussionen um die Form des Buchstabens ß. Wir haben uns dafür entschieden, die Form beizubehalten, die der Autor der Schrift, Jörg Knappen, standardmäßig vorgesehen hat.

Leider hat mein Aufruf nach neuen Beiträgen in der letzten Ausgabe noch nicht zu der gewünschten Resonanz geführt. Aber vielleicht war auch einfach die Zeit zu kurz und viele Mitglieder sind schon fleißig am Schreiben. Das soll aber niemanden davon abhalten, auch zur Feder zu greifen – nein, besser doch zur Tastatur – und einen Beitrag zu verfassen.

Als Beispiel könnte ich mir einen kurzen Tip zu dem aktuellen Thema Euro vorstellen. Dieses Währungszeichen bekommt man bei Einsatz des Paketes `textcomp` mit der Anweisung `\texteuro`. Das Ergebnis ist €. Zu dieser einfachen Möglichkeit gibt es einige Alternativen, teilweise als METAFONT- und teilweise in Form eines PostScript-Type1-Fonts, die man vorstellen und gegeneinander abwägen könnte.

Einige Themen wurden auch der Redaktion als Wunsch zugetragen oder stehen schon länger auf dem Wunschzettel der Redaktion. Wer also etwas für DANTE e.V. und „Die TEXnische Komödie“ tun will, sollte sich bitte an die Redaktion wenden.

Ihr Gerd Neugebauer

Hinter der Bühne

Vereinsinternes

Grußwort

Liebe Mitglieder,

uns liegen lange Grußworte nicht so recht, deshalb wollen wir gleich in *medias res* gehen: Wir sind zwar noch mitten in der „Lernphase“, es zeichnen sich aber jetzt schon Sachverhalte in der Arbeit ab, die uns dazu veranlassen, eine größere Änderung vorzuschlagen. Es wird immer deutlicher, daß der Bereich „Unterstützung der Mitglieder“ in letzter Zeit sträflich vernachlässigt wurde. Im Moment gehen alle Anfragen entweder im Büro oder bei Günter Partosch, der E-Mails an `dante@dante.de` bearbeitet, ein. Sie werden dann, wenn sie nicht direkt beantwortet werden können, teilweise sehr mühsam Problemkreisen zugeordnet und innerhalb des Präsidiums weitergeleitet. Um diesen Prozeß zu vereinfachen und die Beteiligten zu entlasten, möchten wir einen „Beraterkreis“, so der momentane Arbeitstitel, ins Leben rufen.

Deshalb möchten wir alle ansprechen, die etwas zur Unterstützung der Mitglieder beitragen können. Ziel der Idee ist es, für möglichst viele Belange im Zusammenhang mit TEX Ansprechpartner zu haben, die für solche Anfragen zur Verfügung stehen. Keiner muß ein Wizard oder Guru sein! Zu dem Themenkreis gehören natürlich unter anderem die Distributionen für die unterschiedlichen Plattformen, aber auch Probleme und Fragen bei der Anwendung von TEX und LATEX.

Die genaue Organisationsform und die Abwicklung, insbesondere bei brieflichen Anfragen, sind offen und sollen diskutiert werden. Alle, die etwas beitragen können, mögen sich bitte bei uns melden. Wir werden sicher auf der Tagung DANTE'99 in Dortmund Gelegenheit finden, erste konkrete Schritte in einer dann hoffentlich größeren Runde zu besprechen.

Und es kommt noch eine Veränderung auf uns zu: In naher Zukunft werden sich die Adressen einiger E-Mail-Kommunikationslisten ändern. Insbesondere soll die vereinsinterne Diskussionsliste `dante-ev` vom Heidelberger Universitätsrechenzentrum, wo sie bisher beheimatet war, auf unsere eigene Domain

umziehen. Die konkreten Veränderungen und Maßnahmen werden natürlich rechtzeitig bekanntgegeben.

Zum Schluß laden wir nochmals alle herzlich ein, zur Tagung und Mitgliederversammlung in Dortmund zu kommen. Aus Anlaß des zehnjährigen Jubiläums von DANTE e.V. gibt es nicht nur ein hochinteressantes Programm, lassen Sie sich überraschen!

In diesem Sinne verbleiben wir

Thomas Koch (Präsident) Volker RW Schaa (Vizepräsident)

Ergänzungen zum Lizenzabkommen für WinEdt

Volker RW Schaa

Liebe Mitglieder,

die Möglichkeit der Lizensierung von WinEdt über DANTE e.V., die ich in der letzten Ausgabe der Mitgliederzeitung angekündigt habe, wurde dankbar angenommen, so daß bis Ende Januar '99 mehr als 60 Lizenzen beantragt und verteilt wurden.

Leider sind mir in „Die TEXnische Komödie“ 4/98 zwei Fehler unterlaufen, die ich hier korrigieren möchte. Zum einen war die E-Mail-Adresse von Herrn Aleksander Simonic falsch. Die korrekte Adresse lautet `winedt@istar.ca`. In diesem Zusammenhang ist für die Benutzer mit E-Mail-Anschluß sicherlich noch folgendes von Interesse:

Es existiert eine Mailing-Liste in englischer Sprache, über die Sie direkt eine Hilfestellung bei Problemen bekommen können. Der Eintrag zum Abonnieren (`subscribe`) erfolgt an die Adresse `winedt+list-subscribe@wsg.net`, eine eigene Anfrage geht an `winedt+list@wsg.net`. Zum Abbestellen (`unsubscribe`) richten Sie Ihre Mail an `winedt+list-unsubscribe@wsg.net`. Falls Sie erfahren wollen, welche Beiträge bisher über die Liste verteilt wurden, schicken Sie eine Mail an `winedt+list-index@wsg.net`. Das weitere Vorgehen wird in der Antwort beschrieben. Die Mails zum Abonnieren, Abbestellen und zur Index-

Abfrage können leer sein, ihr Inhalt wird immer ignoriert. Wichtig sind die unterschiedlichen Adressaten.

Der Eintrag in diese Mailing-Listen kann auch über die Homepage von Aleksander Simonic erfolgen:

`http://home.istar.ca/~winedt/`

Über diese Adresse können Sie auch weitere Tools, Informationen und interessante Add-Ons bekommen. Eine neuere Version von WinEdt und zugehöriger Software wird erst mit der geplanten DANTE-CTAN-CD-ROM im Frühjahr '99 verteilt werden.

Der zweite Fehler betraf die Dauer der Lizenz. Ich hatte fälschlicherweise einen Passus aus den Vertragsverhandlungen mit Herrn Simonic übernommen, der sich aber auf eine „site-license“ bezog. Bei der Lizenz, die ein Mitglied von DANTE e.V. erhält, gibt es *keine* Begrenzung der Gültigkeitsdauer!

Bretter, die die Welt bedeuten

Verkleinerte und vergrößerte Ausgabe mit LaTeX

Markus Kohm

Eine immer wiederkehrende Frage zu LaTeX betrifft die korrekte Vorgehensweise zur Vergrößerung oder Verkleinerung von Dokumenten. Da diese Frage unterschiedlich motiviert sein kann und der Fragesteller sich oftmals über seine Motive nicht vollständig klar ist, fällt es den Experten häufig nicht leicht, die Frage in der Hinsicht korrekt zu beantworten, daß der tatsächlich gewünschte Effekt erzielt wird.

Im nachfolgenden Artikel werden deshalb ausgehend von einem häufig anzutreffenden Fallbeispiel verschiedene Möglichkeiten vorgestellt und gegeneinander abgewogen. Aufmerksame Leser der deutschsprachigen DE-TeX-/DANTE-FAQ [**?**] wird es nicht verwundern, zumindest eine der hier empfohlenen Lösungen auch dort in Kurzform wiederzufinden.

Vorbemerkung

Im Gegensatz zum sonst üblichen Vorgehen findet sich die Quintessenz dieses Artikels nicht im allerletzten Abschnitt. Um den Anfänger nicht mit Grundlageninformationen vom Hauptthema abzulenken, wird im Hauptteil vieles nur leicht angeschnitten. Für den interessierten Leser finden sich genauere Informationen dann ab dem Abschnitt „Genau gesagt“ auf Seite 20.

Ebenso sind besondere Problemfälle nicht im Hauptartikel aufgeführt. Diese werden stattdessen im Abschnitt „Problemkinder“ auf Seite 15 vorgestellt. Da diese Kenntnisse jedoch für die vorgestellten Lösungswege nicht unbedingt erforderlich sind, finden sich die eigentlichen Lösungen bereits vor diesen Abschnitten.

Das Problem der Vergrößerung

Der häufigste Grund, der eine Vergrößerung notwendig macht, liegt darin, daß für ein erstelltes Werk im Zielformat DIN A5 eine Druckerei die Druckvorlage

nicht in diesem Zielformat, sondern auf DIN A4 vergrößert geliefert bekommen will. Dies ist dann meist für die Herstellung des Druckmediums – beispielsweise eines Films – notwendig, um die gewünschte Güte oder Auflösung zu erreichen.

Das DIN-A-Format und das übereinstimmend definierte ISO-A-Format entsteht durch fortgesetztes Halbieren aus der Grundgröße A0. Von dieser Definition abgeleitet kann man auch festlegen, daß das nächstkleinere Format jeweils durch Division der Seitenlängen mit $\sqrt{2}$ gewonnen wird (siehe Abschnitt „Genau gesagt“, „Papierformate aus dem ABC“, Seite 20).

Umgekehrt entspricht die Vergrößerung von DIN A5 auf DIN A4 also zunächst einer Verdopplung der Papiergröße beziehungsweise einer Vergrößerung aller Längen um den Faktor $\sqrt{2}$. Die anschließende optische Verkleinerung in der Druckerei führt dann wieder zu einer Verkleinerung aller Längen um denselben Faktor. Da diese Verkleinerung optisch erfolgt, ergibt jeder Druckpunkt der Vorlage auch einen Punkt im Druckbild. Damit wird sowohl die horizontale als auch die vertikale Auflösung ebenfalls um den Faktor $\sqrt{2}$ erhöht. Dies ist letztlich der Zweck der ganzen Prozedur. Die Frage bleibt, wie man die Vergrößerung bei Vorlagen, die mit TeX oder LaTeX erstellt werden, am besten erreicht.

Die grundsätzliche Idee

Als erstes könnte man natürlich die Vorlagen direkt in der von der Druckerei verlangten Größe DIN A4 erstellen. Dabei ergibt sich aber beispielsweise das Problem, daß mit entsprechend großer Grundschrift gearbeitet werden müßte. Davon abgesehen, daß es zu den Standardklassen keine Schriftgrößenoptionen `14pt` oder größer gibt, wie man sie für Zielgrößen von 10 pt und mehr benötigte, wäre es keineswegs gesagt, daß solche Dateien nach der Verkleinerung in der Druckerei zum gewünschten Ergebnis führten.

Unter typographischen Gesichtspunkten spielt neben den Aspekten, die den Satzspiegel betreffen (siehe Abschnitt „Genau gesagt“, „Vergrößerung oder Verkleinerung eines Satzspiegels“, Seite 21), die Wahl des Zeichensatzes eine entscheidene Rolle. Dazu muß man wissen, daß eine entsprechend vergrößerte 10 pt-Schrift noch lange keine 14 pt-Schrift und umgekehrt eine verkleinerte 14 pt-Schrift keine 10 pt-Schrift ist. Verdeutlicht wird dies in Abbildung 1, die in der Mitte ein auf 24,88 pt vergrößertes „W“ aus einer 5 pt-Schrift und links davon ein unvergrößertes „W“ direkt aus einer 24,88 pt-Schrift zeigt. Wollte man also für die Vergrößerung statt einer 10 pt-Grundschrift einfach eine 14 pt-Grundschrift verwenden, so wäre das Ergebnis nach der später wieder erfolgenden Verkleinerung nicht das tatsächlich gewünschte.

24,88pt-Schrift in Originalgröße | 5pt-Schrift von METAFONT vergrößert | 5pt-Schrift vom Drucker vergrößert

Abbildung 1: Der Buchstabe „W“ in der Größe 24,88 pt

Um sowohl im Endergebnis den korrekten Zeichensatz als auch einen korrekten Satzspiegel zu erhalten, ist zu empfehlen, die Vorlage zunächst im Zielformat A5 zu erstellen und lediglich für den Ausdruck zu vergrößern. Das hat darüber hinaus den Vorteil, daß der Autor sich nicht mit der Frage beschäftigen muß, wie das Ganze wohl aussieht, wenn es schließlich in der Druckerei verkleinert wurde. Stattdessen kann er die Wirkung direkt an seinen unvergrößerten Probeausdrucken begutachten.

Die unterschiedlichen Möglichkeiten der Vergrößerung

Die Vergrößerung selbst kann auf unterschiedlichste Weise stattfinden. Als erste Idee könnte man am Drucker vielleicht eine andere Auflösung einstellen als die, in der tatsächlich gedruckt wird. Stellt man bei einem Drucker beispielsweise als Auflösung 212 dpi ein und füttert ihn anschließend mit Pixeldaten, die eigentlich für 300 dpi bestimmt waren, so wird das Druckergebnis um den gewünschten Faktor $300/212 \simeq \sqrt{2}$ vergrößert. Allerdings verringert man dabei die tatsächliche Auflösung im Gegenzug um eben diesen Faktor auf die Druckerauflösung von 212 dpi. Ergebnis solchen Handelns ist, daß das Ziel der Auflösungserhöhung in der Druckerei zunichte gemacht wird. Das rechte „W“ in Abbildung 1 demonstriert dies, wobei der Unterschied in der Auflösung zwischen dem von METAFONT und dem vom Drucker vergrößerten Buchstaben von 1270 dpi zu 254 dpi (= 1270 dpi/5) aufgrund der hohen Auflösung nicht unbedingt sofort ins Auge springt.

Die Methode, die Druckerauflösung zu ändern, ist meist nicht ganz einfach zu beschreiten, da man irgendwie den Datenstrom vom DVI-Treiber zum Drucker manipulieren muß. Werkzeuge hierfür sind mir nur für PostScript bekannt. Außerdem bieten Drucker im allgemeinen nicht beliebige Auflösungen.

DVI-Treiber verfügen allerdings über eine Möglichkeit, auf die Größe oder Vergrößerung Einfluß zu nehmen. Zur Erklärung dieser Option muß ein wenig ausgeholt werden.

In einer DVI-Datei sind Größeninformationen nicht in *dots per inch* (dpi), sondern in *units* abgelegt. Die Größe einer *unit* selbst wird durch den Quotienten

aus zwei ganzzahligen Konstanten und einen zusätzlichen Faktor definiert, die im Kopf der DVI-Datei abgelegt sind. Dieser zusätzliche Faktor ist abgekürzt mit *mag* für das englische *magnification*, zu Deutsch *Vergrößerung*, bezeichnet und wird in Promille angegeben. Normalerweise ist dieser Wert auf 1000, was keiner Vergrößerung oder Verkleinerung entspricht, voreingestellt (siehe auch „Grundlage“, „Größenangaben in DVI-Dateien“, Seite 22).

Es bietet sich fast von selbst an, für Größenmanipulationen nicht die erwähnten ganzzahligen Konstanten, sondern eben die Vergrößerungsvariable *mag* zu verwenden. Damit sind dann Vergrößerungen und Verkleinerungen auch eindeutig als solche erkennbar, wohingegen eine Veränderung des anderen Wertes ihre Ursache auch darin haben können, daß mit einer anderen Basiseinheit gearbeitet wurde. Dies ist beispielsweise bei der DVI-Ausgabe von groff [**?**] der Fall. Bei groff handelt es sich um ein Dokumentformatierungssystem, das verschiedene Ausgabeformate beherrscht und hauptsächlich für Programmanleitungen im Unix-Bereich verwendet wurde.

Zu betrachten wäre dann noch, ob bei dieser Art der Vergrößerung auch der richtige Font geladen wird. Eine Fontdefinition in der DVI-Datei besteht unter anderem aus dem Namen des Fonts und seiner Designgröße [**?**]. Die Designgröße ist ebenfalls in der Einheit *unit* angegeben. Vom DVI-Treiber wird immer nach dem Zeichensatz in der angegebenen Designgröße gesucht (siehe Abschnitt „Genau gesagt“, „Wie ein DVI-Treiber einen Zeichensatz findet“, Seite 23).

Eine Veränderung von *mag* bewirkt also über die Veränderung der *unit* auch, daß andere Zeichensatzdateien geladen werden. Wichtig ist festzuhalten, daß vorhandene Zeichensatzdateien in Pixeldarstellung nicht auf eine andere Größe umgerechnet werden. Stattdessen werden Zeichensätze, die in der entsprechenden Vergrößerung oder Verkleinerung – im allgemeinen von METAFONT – berechnet wurden, angefordert.

Deshalb bieten die DVI-Treiber auch die Möglichkeit, den *mag*-Wert in der DVI-Datei durch einen anderen Wert zu ersetzen oder mit einem zusätzlichen Wert zu multiplizieren. Die Nutzung dieser Möglichkeit führt dann dazu, daß sämtliche Ausgaben entsprechend größer oder kleiner werden.

Aber nicht nur die DVI-Treiber kennen die Variable *mag*. TeX selbst bietet diese Variable ebenfalls an. Mit der folgenden Zuweisung in der Präambel eines Dokuments würde beispielsweise eine Vergrößerung um den gewünschten Faktor $\sqrt{2}$ erreicht.

```
\mag=1414
```

```
\documentclass{minimal}
\begin{document}
  \fontsize{10}{12}\selectfont W
\end{document}
```

Abbildung 2: Die Referenzdatei zur Ermittlung der verwendeten Fonts

```
This is dvips 5.58 Copyright 1986, 1994 Radical Eye Software
' TeX output 1997.11.21:1050' -> refdatei.ps
Defining font () cmr10 at 10.0pt
Loading pk font cmr10.300pk at 10.0pt
<texc.pro>. [1]
```

Abbildung 3: dvips-Ausgabe zu der Referenzdatei aus Abbildung 2

Dies ist auch in LaTeX zulässig. Da der Wert, der `\mag` zugewiesen wird, direkt in den Kopf der DVI-Datei geschrieben wird und sich somit auf das gesamte Dokument auswirkt, sollte die Zuweisung ebenfalls im Kopf des Dokuments, also in jedem Fall vor `\begin{document}` erfolgen. Es ist zulässig und empfehlenswert, diese noch vor der Auswahl der Dokumentklasse vorzunehmen.

Die beiden relevanten Unterschiede zwischen der Veränderung von *mag* über eine Option beim Aufruf des DVI-Treibers und der Zuweisung an `\mag` im Dokumentkopf sind in den Abschnitten „Problemkinder“, „Die true-Einheiten“ und „Der Nullpunkt einer Seite“ ab Seite 17 zu finden.

Vergrößerung in der Anwendung

In diesem Abschnitt werden nun einige Beispiele für die Vergrößerung demonstriert. Dazu wird die Referenzdatei aus Abbildung 2 auf unterschiedliche Art und Weise vergrößert. Die Referenzdatei erzeugt nur einen einzelnen Buchstaben auf dem Papier. Es wird kontrolliert, welcher Font dafür verwendet wird. Als Referenz-DVI-Treiber wird dvips verwendet. Diesen kann man mit der Option `-d 4` dazu veranlassen auszugeben, welchen Font er gerade lädt. Abbildung 3 zeigt die zur Referenzdatei gehörende Ausgabe von dvips.

Uns interessiert an dieser Stelle aus der Ausgabe von dvips nur die hervorgehobene Zeile. Diese bedeutet, daß der Zeichensatz `cmr10` in der Auflösung `300` dpi für die Schriftgröße `10.0pt` geladen wird. 300 dpi ist hierbei die voreingestellte Auflösung des Druckers. Die `10` im Fontnamen `cmr10` steht ebenfalls für 10 pt, die Entwurfsgröße der Schrift. Es wird also der Font cmr in der Ent-

wurfsgröße 10 pt in der Größe 10 pt und der Druckerauflösung 300 dpi geladen, was keiner Vergrößerung entspricht. Verwendet man statt der OT1- die T1-Codierung und damit die EC-Fonts, so wird stattdessen `ecrm1000.300pk at 10.0pt` geladen. Hier steht `1000` im Namen nicht etwa für die Entwurfsgröße 1000 pt, sondern für 10,00 pt, da bei den EC-Fonts ein anderes Namensschema verwendet wird als bei den CM-Fonts.

Ergänzen wir nun die Präambel obiger Grunddatei um eine Zeile für die geforderte Vergrößerung um den Faktor $\sqrt{2}$.

```
\mag=1414 % Vergrößerung um ca. Wurzel 2
```

Daraufhin gibt dvips an, daß nun `cmr10.424pk at 10.0pt` geladen wird. Dies bedeutet also, daß der Font cmr in der Entwurfsgröße 10 pt in der Auflösung 424 dpi und der Größe 10 pt verwendet wird. Da die Ausgabe auf dem Drucker jedoch nicht mit 424 dpi sondern mit 300 dpi erfolgt, entspricht dies einer Vergrößerung um den Faktor $424/300 \simeq \sqrt{2}$.

Verdeutlicht wird die unterschiedliche Behandlung von CM- und EC-Schriften noch, wenn man explizit eine Schrift in 14,40 pt-Größe anfordert. Dies ist möglich, indem man in der Referenzdatei den Wert 10 im Größenauswahlbefehl `\fontsize{10}{12}` durch `14.40` ersetzt. Man erhält dann je nach Codierung als Ergebnis beim Lauf von dvips eine der folgenden Zeilen.

```
Loading pk font cmr12.360pk at 14.4pt
Loading pk font ecrm1440.300pk at 14.4pt
```

Bei CM wird also eine vergrößerte Schrift benutzt, wenn auch nicht cmr10 sondern cmr12, also ausgehend von der Entwurfsgröße 12 pt, während bei EC die entsprechende Entwurfsgröße angefordert wird. Ganz nebenbei sind wir so auf einen interessanten Vorteil der EC-Schriften gegenüber den CM-Schriften gestoßen: es existieren wesentlich mehr Entwurfsgrößen. Übrigens wären die oben angegebenen Zeichensätze genau diejenigen, die angefordert würden, wenn wir gemäß Problemstellung nicht in A5, sondern in A4 arbeiten würden. Bei der Verkleinerung in der Druckerei würde so ein verkleinerter 12 pt- beziehungsweise 14 pt-Zeichensatz entstehen und nicht der gewünschte 10 pt-Zeichensatz. Wir sehen also, daß die Entscheidung, in der Zielgröße zu arbeiten, vorteilhaft war.

Es stellt sich nun die Frage, welches Ergebnis wir erhalten, wenn wir dvips per Option `-x 1414` sagen, daß die Ausgabe der Grunddatei um 1,414 vergrößert werden soll. Es ist nicht sonderlich verwunderlich, daß das Ergebnis exakt das gleiche wie bei Angabe eines entsprechenden `\mag`-Wertes in der Präambel ist, da beide Vorgehensweisen zu einer entsprechenden Änderung des *mag*-Wertes führen.

Sollgröße:	10pt	14pt	14pt	14pt
`\mag`:	1000	1414	1440	1000
Zeichensatz:	cmr10	cmr10	cmr10	cmr12
Auflösung:	254 dpi	359 dpi	366 dpi	305 dpi
Beispiel:				
Zeichensatz:	ecrm1000	ecrm1000	ecrm1000	ecrm1440
Auflösung:	254 dpi	359 dpi	366 dpi	254 dpi
Beispiel:				

Tabelle 1: Zusammenstellung der verwendeten Vergrößerungen und Zeichensätze bei einer Druckerausgabe mit 254 dpi

Tabelle 1 zeigt eine Zusammenfassung der Beispiele aus diesem Kapitel. In den Spalten steht jeweils untereinander als oberstes die Sollgröße und darunter der *mag*-Wert, dann jeweils für OT1- und T1-Codierung der verwendete Zeichensatz, darunter die daraus und aus dem *mag*-Wert resultierende Auflösung ausgehend von einer Druckerauflösung von 254 dpi und darunter jeweils ein Buchstabenbeispiel, wie es mit Hilfe der Referenzdatei erstellt worden ist. Um das Raster der Buchstaben sichtbar zu machen, wurden sie in einer geringeren Auflösung erzeugt und zusätzlich vergrößert. Es entstanden so doppelt große Buchstaben in einem Zehntel der Auflösung der vorliegenden Zeitschrift „Die TEXnische Komödie“ von 1270 dpi. Die neben der um $\sqrt{2}$ angegebene Vergrößerung um den Faktor 1,440 spielt im Abschnitt „Genau gesagt“, „Gute Ergebnisse mit weniger Zeichensatzberechnungen“ ab Seite 25 eine Rolle.

Ergebnismanipulation

Nachdem dvips schon diverse PostScript-Dateien erstellt hat, liegt die Idee nahe, statt der DVI-Datei doch die PostScript-Ausgabe der Grunddatei direkt mit PostScript-Mitteln zu vergrößern. Diese Idee ist weder abwegig noch schwer umzusetzen, gibt es doch in den PS-Utilities [**?**] genau die richtigen Werkzeuge für allerlei Manipulation an PostScript-Dateien.

Wissen muß man nun allerdings, daß dvips die Daten aus den oben bezeichneten pk-Dateien in entsprechend umgewandelter Form als Fonts in die PostScript-Ausgabe einbindet. Diese pk-Dateien enthalten keine Vektor- sondern nur Bitmap-, also Pixelinformationen der Fonts. Die von dvips eingebundenen Fonts sind also auch Bitmap-Fonts. Vergrößerung oder Verkleinerung von Pixelbildern führt aber immer zu erheblichen Qualitätsverlusten. Dieser Weg, der für das rechte „W“ in Abbildung 1 verwendet wurde, ist also zunächst abzulehnen. Anders sieht es aus, wenn statt der Bitmap-Fonts PostScript-Fonts im Type-1-Format eingebunden werden. Von den CM-Fonts selbst gibt es beispielsweise auch mehrere PostScript-Versionen im Type-1-Format [**?**, **?**], die jedoch im Falle der BaKoMa-Font-Collection gewissen Copyright-Bedingungen unterliegen und auch nicht immer vollständig sind. Dem PostScript-Treiber dvips kann man per Option bzw. Fontmap-Datei beibringen, solche Fonts zu verwenden. Allerdings vergrößern sich dabei die erzeugten PostScript-Dateien erheblich.

Mit Hilfe von dvips wäre es auch möglich, andere PostScript-Fonts zu verwenden. Leider haben PostScript-Schriftfamilien wie Times den Nachteil, daß sie in der Regel nur in einer Entwurfsgröße vorliegen. Werden andere Größen benötigt, so entstehen diese dann wiederum durch einfache optische Skalierung der vorhandenen Grundgröße. Daß dieses Vorgehen nachteilig und typographisch bedenklich ist, wurde bereits im Abschnitt „Die grundsätzliche Idee“ auf Seite 8 erläutert und durch Abbildung 1 verdeutlicht. Gerade die Verfügbarkeit diverser Entwurfsgrößen, insbesondere der CM- und EC-Standardzeichensätze und deren automatische Auswahl stellt einen entscheidenden Vorteil von LaTeX dar. Man sollte ihn deshalb nicht ohne guten Grund aufgeben.

Verwendet man dvips mit Bitmap-Fonts, ist außerdem zu beachten, daß diese in der Auflösung desjenigen Druckers erstellt wurden, mit dem der Ausdruck dann auch tatsächlich stattfinden soll. Ist dies nicht der Fall, müßten die Zeichensätze wiederum als Bitmaps verkleinert oder vergrößert werden, was ja nicht ohne, teilweise erhebliche, Qualitätsverluste möglich ist. Neben der Auflösung gibt es weitere Geräteeigenschaften zu beachten. All diesen Faktoren wird bei der Fontberechnung durch METAFONT mit Hilfe des Geräteparameters `mode` Rechnung getragen. Nähere Erklärungen hierzu sind [**?**] und der Geräteparameterdatei `modes.mf` zu entnehmen.

Nebenbei haben wir also gelernt, daß dvips die Bitmap-Fonts im *Modus* des Zieldruckers benötigt. Dies ist auch dann der Fall, wenn eine Druckerei direkt mit PostScript-Dateien beliefert werden kann. Gegebenenfalls muß man also in der Druckerei nachfragen, mit welchem Gerät und in welcher Auflösung die Ausgabe erfolgt.

Zwischenfazit

Geht es darum, Druckvorlagen in einer anderen als der Zielgröße zu erstellen, damit bei der optischen Verkleinerung auf die Zielgröße eine höhere Auflösung erreicht wird, so stellt die Manipulation des Vergrößerungsfaktors *mag* der DVI-Datei einen Weg dar, der auf jeden Fall zum gewünschten Ergebnis führt. Ob die Manipulation direkt in der TeX-Datei über eine Zuweisung an `\mag` oder durch eine Option beim Aufruf des DVI-Treibers erfolgt, ist dabei hauptsächlich eine Frage der Handhabung. Mir persönlich erscheint die Verwendung einer DVI-Treiber-Option praktischer, da ich so alle Probeausdrucke ohne zusätzliche Angabe in der Zielgröße erhalte und ohne Änderung der Dokumentdatei jederzeit auch Druckvorlagen erstellen kann.

Demgegenüber führt die nachträgliche Manipulation der PostScript-Datei nicht immer zu einem optimalen, sondern teilweise auch zu einem dem eigentlichen Ziel entgegenstehenden, nämlich in der Auflösung nicht verbesserten, Ergebnis. Die direkte Dokumenterstellung in einem anderen als dem Zielformat, also insbesondere in der von der Druckerei gewünschten Größe, scheidet aus typographischen Gründen von vornherein aus.

Problemkinder

Eigentlich sollte man denken, wir wären nun am Ziel angelangt und könnten als Ergebnis festhalten, daß eine Vergrößerung oder Verkleinerung sehr einfach entweder per `\mag`-Zuweisung in der Präambel oder per DVI-Treiber-Option realisiert werden kann. Leider gibt es ein paar Problemfälle, die in den nachfolgenden Abschnitten kurz angesprochen werden sollen. Teilweise ist dafür auch das tiefere Verständnis aus dem Abschnitt „Genau gesagt“ ab Seite 20 von Vorteil.

Bilder in Fremdformaten

Werden Bilder nicht mit TeX- oder LaTeX-Mitteln erstellt, sondern in Fremdformaten realisiert, so findet bei deren Ausgabe nicht zwangsläufig eine Umrechnung der Größe statt. Ursache dafür ist, daß diese Bilder mit Hilfe der `\special`-Anweisung eingebunden werden. `\special`-Anweisungen werden als unveränderte Textinformation in die DVI-Datei geschrieben. Angenommen, die Anweisung `\special{hline 10cm}` würde von einem DVI-Treiber als Linie der Länge 10 cm interpretiert. Im Falle einer Vergrößerung würde in der DVI-Datei nach wie vor „`hline 10cm`“ stehen, da die textuelle Information „`10cm`“ von TeX unberührt bleibt. Der DVI-Treiber müßte also wissen, daß

```
\begin{filecontents*}{a.ps}
%!
%%BoundingBox:100 100 172 172
100 100 moveto
72 72 rlineto 72 neg 0 rlineto 72 72 neg rlineto stroke
/Times-Roman findfont 72 scalefont setfont (A) show
showpage
\end{filecontents*}
\documentclass{article}
\usepackage[dvips]{graphics}
\pagestyle{empty}
\begin{document}
  \fbox{\includegraphics{a.ps}}
\end{document}
```

Abbildung 4: Beispieldatei zur Vergrößerung von PostScript-Bildern

eine Vergrößerung oder Verkleinerung vorliegt, um die Linie gegebenenfalls in der Größe zu ändern. Analoges gilt, wenn andere Zahlenwerte, beispielsweise zur Beschreibung der Ausmaße eines PostScript-Bildes, angegeben werden.

Wie wir jedoch festgestellt haben, stellt eine vom Standardwert 1000 abweichende *mag*-Angabe im Kopf der DVI-Datei ein eindeutiges Indiz für eine Vergrößerung bzw. Verkleinerung dar. Die Frage ist, ob dieses von DVI-Treibern auch tatsächlich entsprechend interpretiert wird.

Tatsächlich haben wir Glück und sowohl bei einer Vergrößerung per `\mag=1414` als auch bei einer Vergrößerung per dvips-Option wird eine nachgeladene PostScript-Datei mit vergrößert, wie ein Experiment mit einer Testdatei aus Abbildung 4 zeigt. Diese Datei erzeugt zunächst mit Hilfe der `filecontents`-Umgebung die PostScript-Datei `a.ps` und bindet diese dann mit Hilfe des Pakets `graphics` in das LaTeX-Dokument ein. Vergrößert man dieses Dokument entweder durch eine zusätzliche Zeile `\mag=1414` oder durch Angabe der Option `-x 1414` beim Aufruf von dvips, so wird die Graphik ebenfalls vergrößert. Abbildung 5 zeigt das unvergrößerte Ergebnis. Die PostScript-Datei `a.ps` ist übrigens der Anleitung zum Paket graphics entnommen.

Damit ist also auch im Falle der Vergrößerung oder Verkleinerung die Einbindung von Bildern und Graphiken zumindest als PostScript-Datei unproblematisch.

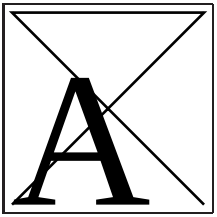

Abbildung 5: Das unvergrößerte Ergebnis der Beispieldatei aus Abbildung 4

Die *true*-Einheiten

In TeX und LaTeX ist es möglich, Größenangaben dadurch von Vergrößerung und Verkleinerung auszunehmen, daß man der Einheit ein `true` voranstellt. Man schreibt dann beispielsweise `12truemm` an Stelle von `12mm`. Allerdings betrifft dies nur TeX-interne Vergrößerung und Verkleinerung durch Zuweisungen an `\mag`. Die Frage ist, in welcher Weise diese Größen vor Beeinflussung durch den geänderten *mag*-Wert ausgenommen werden.

TeX selbst stellt sicher, daß alle *true*-Einheiten von der Vergrößerung oder Verkleinerung unberührt bleiben, indem diese entsprechend dem veränderten *mag*-Wert umgerechnet werden. Die DVI-Dateien kennen keine *trueunit*, also keine Größenangaben, auf die der *mag*-Wert im Kopf der DVI-Datei nicht anzuwenden ist. Anders gesagt, der *mag*-Wert muß auf alle Größenangaben in den DVI-Dateien angewendet werden.

Zur Verdeutlichung werden im Beispiel aus Abbildung 6 zwei Längenvariablen mit den Namen `\NormaleLaenge` und `\TrueLaenge` definiert. Beiden wird als Länge 1414 pt zugewiesen, der `\NormaleLaenge`-Variable in gewohnter Weise, der Variable `\TrueLaenge` jedoch als 1414 truept. Die Gesamtausgabe wird außerdem per `\mag=1414` um den Faktor 1,414 vergrößert. Desweiteren wird LaTeX in diesem Beispiel dazu veranlaßt, die aktuellen Werte der beiden Variablen nach der Zuweisung auszugeben. Dies geschieht mit Hilfe des Grundbefehls `\the`, der im TeXbook [**?**] beschrieben ist. Eine DVI-Datei wird nicht erzeugt, da für uns nur die Bildschirmausgabe des Makros `\typeout` interessant ist. Weil TeX für die Vergrößerung lediglich den *mag*-Wert im Kopf der DVI-Datei verändert, erwarten wir, daß der Wert von `\NormaleLaenge` unverändert mit 1414 pt angegeben wird. Da die Werte der Längenvariablen mittels `\the` immer in pt und nicht etwa in truept ausgegeben werden, erwarten wir jedoch, daß für `\TrueLaenge` nicht 1414 pt ausgegeben werden, wie dies der

```
\documentclass{minimal}
\mag=1414 % Vergrößerung
% Längenvariablen definieren
\newlength{\NormaleLaenge}
\newlength{\TrueLaenge}
% Werte zuweisen
\setlength{\NormaleLaenge}{1414pt}
\setlength{\TrueLaenge}{1414truept}
% Tatsächlich gespeicherte Werte ausgeben
\typeout{\NormaleLaenge=\the\NormaleLaenge}
\typeout{\TrueLaenge=\the\TrueLaenge}
% Keine DVI-Datei erzeugen
\begin{document}\end{document}
```

Abbildung 6: Beispiel zur Verdeutlichung von *true*-Größen

Fall wäre, wenn wir auf die Vergrößerung mittels `\mag=1414` verzichtet hätten. Tatsächlich werden unter anderem die folgenden beiden Ausgabezeilen erzeugt.

```
\NormaleLaenge=1414.0pt
\TrueLaenge=1000.0pt
```

Demnach wird die `\TrueLaenge` also um den `\mag` zugewiesenen Wert verkleinert, damit bei der anschließenden Vergrößerung – durch den *mag*-Wert 1414 in der DVI-Präambel statt des normalen Wertes 1000 – wieder 1414 pt entstehen. Demgegenüber werden aus den 1414,0 pt von `\NormaleLaenge` infolge der Multiplikation mit *mag*/1000 dann effektive 1999,396 pt.

Ganz anders sieht es aus, wenn die Vergrößerung mit Hilfe einer Option des DVI-Treibers geschieht. Da TeX in diesem Fall keine Informationen besitzt, daß eine Vergrößerung erfolgen soll, werden truept und pt gleich behandelt. Sie bleiben von TeX unverändert. Somit werden identische *unit*-Werte in die DVI-Datei geschrieben. Vom DVI-Treiber selbst sind die Werte nicht mehr unterscheidbar und werden damit beide gleichermaßen vergrößert.

Diese unterschiedliche Behandlung von true-Größen ist im Normalfall einer der signifikanten Unterschiede zwischen einer `\mag`-Zuweisung in TeX bzw. LaTeX und der Angabe einer *mag*-Option beim Aufruf des DVI-Treibers.

Der Nullpunkt einer Seite

Bei DVI-Dateien liegt der Nullpunkt, von dem aus alle Koordinaten berechnet werden, in der linken oberen Ecke jeder Seite. Allerdings ist er aus historischen Gründen nicht beim Zusammentreffen der beiden Papierkanten, sondern jeweils ein Inch von diesen Kanten in das Innere des Papiers verschoben. Dieser Versatz um einen Inch wird laut Festlegung auch bei einer Vergrößerung oder Verkleinerung wie bei einer *true*-Größe nicht verändert. Im Unterschied zu dieser ändert sich der Versatz selbst bei einer Vergrößerung oder Verkleinerung mit Hilfe der Option des DVI-Treibers nicht. Letztlich bedeutet dies, daß ein DVI-Treiber zu den ermittelten und skalierten Koordinaten jeweils ein unskaliertes Inch hinzuaddiert.

Sollen nun die Proportionen der Ränder erhalten bleiben, so muß man die Randeinstellungen korrigieren, indem man die Differenz zwischen diesem konstanten Versatz und einem mitvergrößerten Versatz bei den Randeinstellungen von `\topmargin` und `\oddsidemargin` hinzuaddiert. Vergrößert man ein Dokument durch `\mag`, so kann man die Berechnung LaTeX durch Benutzung der *true*-Größen nach der Zuweisung an `\mag` übertragen:

```
\addtolength{\topmargin}{-1truein}
\addtolength{\topmargin}{1in}
\addtolength{\oddsidemargin}{-1truein}
\addtolength{\evensidemargin}{1in}
```

Wird ein Dokument mit Hilfe der Option des DVI-Treibers vergrößert, so sollte man die Randeinstellungen in LaTeX unverändert lassen und die Korrektur nur über Randeinstellungsoptionen des DVI-Treibers, beispielsweise die Option `-O` bei dvips, angeben. Die Korrekturwerte für den horizontalen sowie vertikalen Versatz des Nullpunkts ergeben sich, angegeben in Inch, aus dem gewählten Vergrößerungsfaktor vermindert um Eins. Bei der Vergrößerung um 1,414 ergibt sich somit eine Korrektur um 0,414 Inch und diese wird durch Angabe der Optionen `-x 1414 -O 0.414in,0.414in` beim Aufruf von dvips angegeben. Arbeitet man häufiger mit Verkleinerungen oder Vergrösserungen, empfiehlt es sich bei der Verwendung von dvips dafür eine Optionendatei anzulegen, in denen sowohl die Option `x` als auch die Option `O` bereits korrekt eingestellt sind. Näheres dazu ist der Anleitung des DVI-Treibers zu entnehmen.

Wir halten fest, daß sich Vergrößerung oder Verkleinerung überhaupt nicht auf die Lage des Nullpunkts auswirken und wir diesen daher entweder über eine Korrektur der Randeinstellungen in TeX bzw. LaTeX oder über eine weitere

Option des DVI-Treibers korrigieren müssen, wobei wir im letzten Falle die Korrekturwerte abhängig vom Vergrößerungsfaktor selbst berechnen müssen.

Spickzettel per \mag?

Im Gegensatz zur bisherigen Problemstellung könnte man sich natürlich auch vorstellen, daß jemand ein Minidokument in A6 und 8 pt-Schrift erstellen will. Da die Standardklassen weder eine Option `a6paper` noch `8pt` bieten, liegt die Idee nahe, stattdessen in A5 mit 11 pt zu arbeiten und die oben genannten Möglichkeiten anschließend zur Verkleinerung zu verwenden. Für eine Verkleinerung gilt nahezu alles bisher festgestellte, handelt es sich dabei doch lediglich um eine Vergrößerung um einen Faktor kleiner als 1, also mit einem *mag*-Wert kleiner 1000. Wer den Artikel bisher gründlich gelesen hat, weiß daher auch, daß dieser Ausweg nur ein Holzweg wäre. Zum einen ist, wie bereits eingangs ausgeführt wurde, ein A5-Satzspiegel nach der Verkleinerung eben noch immer ein verkleinerter A5- und kein A6-Satzspiegel. Zum anderen unterscheidet sich die Entwurfsgröße 8 pt der Schriften eben von einer auf 8 pt verkleinerten Entwurfsgröße mit 11 pt, wie Abbildung 1 überdeutlich demonstriert.

Die Lösung für dieses Problem liegt in der Definition der entsprechenden Papiergröße und Grundschrift, die ich in diesem Artikel jedoch schuldig bleibe. Hinweise hierzu finden sich beispielsweise in der Anleitung für Paketautoren [**?**], die jeder LaTeX-Distribution beiliegt, in der Implementierungsdokumentation der Standardklassen [**?**] sowie in einem früheren Artikel der Zeitschrift „Die TeXnische Komödie“ [**?**].

Genau gesagt

In den folgenden Abschnitten werden Grundlagen behandelt, die für das eigentliche Verständnis des Artikels nicht zwingend erforderlich sind. Sie erschienen mir aber interessant genug, um sie wenigstens kurz anzuschneiden und weiterführende Literatur dazu anzugeben.

Papierformate aus dem ABC

Die A-, B-, C- und D-Formate nach DIN 476 sowie die gleichbezeichneten Papierformate nach ISO 216 sind jeweils ausgehend von den Grundgrößen A0, B0, C0 und D0 definiert. Dabei gilt beispielsweise für A0 folgendes Seitenverhältnis von längerer Seite (H) zu kürzerer Seite (B).

$$\frac{\mathrm{H_{A0}}}{\mathrm{B_{A0}}} = \frac{1189\,\mathrm{mm}}{849\,\mathrm{mm}} = \sqrt{2}.$$

ISO-Format	Breite/mm	Höhe/mm
A0	841	1189
B0	1000	1414
C0	917	1297
D0	771	1090

Tabelle 2: Die Grundgrößen des ISO-A-, -B-, -C- und -D-Formates

Bei diesem Verfahren wird zur Gewinnung des nächst kleineren Papierformats das Papier in der Mitte der Längsseite halbiert. Somit gilt allgemein für das A-Format:

$$\mathrm{H}_{\mathrm{A}(n+1)} = \mathrm{B}_{\mathrm{A}(n)} = \sqrt{2} \cdot \mathrm{H}_{\mathrm{A}(n)} \quad (1)$$

$$\mathrm{B}_{\mathrm{A}(n+1)} = \tfrac{1}{2}\mathrm{H}_{\mathrm{A}(n)} = \sqrt{2} \cdot \mathrm{B}_{\mathrm{A}(n)}. \quad (2)$$

Gerechnet wird dabei mit abgerundeten Millimeterwerten.

Analoges gilt auch für B-, C- und D-Format. Die Grundgrößen sind in Tabelle 2 angegeben.

Vergrößerung oder Verkleinerung eines Satzspiegels

Will man bei der Veränderung des Papierformates die Schriftgröße dazu passend ändern, so ist es aus typographischer Sicht nicht ausreichend, alle Werte in `size10.clo` mit dem entsprechenden Faktor zu multiplizieren, um so beispielsweise zu `size12.clo` zu kommen. Bei der Erstellung der Dateien für die Grundschriftgrößen ist weit mehr als nur einfache Mathematik zu beachten. Der *Graueindruck* einer Seite spielt hierbei eine zentrale Rolle. Dieser entsteht, wenn man eine Seite auf gewisse Entfernung unscharf betrachtet. Der Graueindruck soll möglichst gleichmäßig sein. Dadurch kann es je nach Schriftbild und Schriftgröße beispielsweise notwendig werden, den Zeilenabstand größer oder kleiner zu wählen.

Aber auch die Verteilung von Text und Rändern ist wichtig. Zwar kann bei einer insgesamt proportionalen Größenänderung die optimale Zeilenlänge beibehalten werden. Dabei blieben aber Grenzwerte, die ebenfalls wichtig sind, gänzlich unberücksichtigt. So wird ein zu kleiner Rand rasch als gar nicht mehr existent wahrgenommen. Aber auch rein praktische Gründe spielen eine Rolle. Beispielsweise sollte in einem Buch geblättert werden können, ohne dabei Fingerabdrücke zu hinterlassen. Aufgrund der häufig nicht vollständigen Abriebfestigkeit und Fettbeständigkeit von Druckerschwärze entstehen Fingerabdrücke meist dann, wenn man beim Umblättern oder Halten eines Buches

gezwungen ist, in den Textbereich zu fassen. Ränder werden darüber hinaus unter Umständen vom Leser auch für handschriftliche Anmerkungen verwendet. Ein nur wenige Millimeter breiter Rand ist hierfür nicht zu gebrauchen. So spielt also auch die beabsichtigte Verwendung eines Druckwerks eine Rolle bei der Festlegung des Satzspiegels.

Da ein Rand bei der Verkleinerung aber ebenfalls kleiner wird, erfüllt er die einmal getroffenen Mindestanforderungen anschließend unter Umständen nicht mehr. Hinzu kommt, daß auch diese Mindestanforderungen vom Seitenformat und der Schriftgröße abhängen können. Eine Erhöhung der Seitenlängen des Papiers um einen konstanten Wert muß also nicht zwangsläufig eine Vergrößerung der Ränder um denselben Wert rechtfertigen.

Allein schon aus diesem Grund ist es nahezu zwingend, den Satzspiegel für das Zielformat festzulegen und dann sinnvollerweise auch mit diesem Format zu arbeiten. Eine für einen Zwischenschritt der Produktion erforderliche Größenveränderung sollte sich nicht negativ auf das Endergebnis auswirken, also möglichst nur für diesen einen Schritt erfolgen.

Größenangaben in DVI-Dateien

Wie bereits erwähnt, werden alle Größenangaben in DVI-Dateien in *units* angegeben. Die Größe einer *unit* ist dabei nicht in der Definition des DVI-Formats festgelegt, sondern ist eine variable Größe. Sie hängt unter anderem von dem Programm ab, das DVI-Ausgaben tätigt und wird im Kopf der DVI-Datei abgelegt [**?**]. Um sich auf Ganzzahlberechnungen beschränken zu können, erfolgt die Speicherung in Form eines Bruches zweier natürlicher Zahlen, *num* und *den*. Diese beiden Abkürzungen stehen für die englischen Bezeichnungen *numerator* (Zähler eines Bruchs) und *denominator* (Nenner eines Bruchs). Der so entstehende Wert in $10^{-7}\,\mathrm{m}$ wird außerdem noch mit dem Tausendstel des Vergrößerungsfaktors *mag* multipliziert. Dieser Variablenname ist die Abkürzung des englischen Wortes *magnification* (Vergrößerung). Damit gilt also:

$$unit = \frac{mag}{1000} \cdot \frac{num}{den} \cdot 10^{-7}\,\mathrm{m}.$$

Die Werte für *num* und *den* werden bei TeX-Ausgaben durch die Grundeinheit sp (*scaled point*) bestimmt, in der TeX rechnet [**?**].

$$1\,\mathrm{sp} = 1/65536\,\mathrm{pt} = 254/473628672\,\mathrm{cm} = 25400000/473628672 \cdot 10^{-7}\,\mathrm{m}$$

Damit ist $num = 25400000$ und $den = 473628672$. Im Normalfall, also ohne Vergrößerung und Verkleinerung, ist $mag = 1000$.

Will man die Ausgabe vergrößern oder verkleinern, so muß man lediglich eine der drei Variablen entsprechend verändern. Um Vergrößerungen und Verkleinerungen eindeutig identifizieren zu können, ist es sinnvoll, nicht *num* oder *den*, sondern ausschließlich *mag* für diesen Zweck zu verwenden.

Die maximale Länge, die TeX in eine DVI-Datei schreiben kann, ist 2^{31} sp $\simeq$ 11,5 m. Mit anderen Programmen und anderen Werten für *num* und *den* gelten andere Maximalwerte.

Wie ein DVI-Treiber einen Zeichensatz findet

Generell sind Verzeichnis und Name einer Zeichensatzdatei, wie sie DVI-Treiber verwenden, sowohl vom System als auch vom DVI-Treiber selbst abhängig. Bei Verwendung von pk-Fonts gibt es jedoch ein paar grundsätzliche und ein paar häufige Gemeinsamkeiten.

Zum Laden eines Fonts rechnet der DVI-Treiber die Designgröße, die in der DVI-Datei in *units* angegeben ist, im allgemeinen in dpi um und gewinnt den Dateinamen dann aus dem Font-Namen und dieser Größe in dpi. Der Font-Name ist normalerweise identisch mit dem Namen der entsprechenden tfm-Datei ohne deren Endung „tfm“. Die tfm-Datei enthält die sogenannte Fontmetrik. Sie ist alles, was TeX selbst über den Font an Informationen benötigt. Vom DVI-Treiber wird sie normalerweise ebenfalls verwendet.

Der dpi-Wert wird dabei auf ganze dpi gerundet. Leider verwenden unterschiedliche Treiber hierzu teilweise unterschiedliche Rundungsverfahren. Je nach System wird dann beispielsweise bei einer Druckerauflösung von 300 dpi und ohne Vergrößerung oder Verkleinerung die Datei `FONT0300\cmr10.pk` oder die Datei `cmr10.300pk` geladen. Vereinzelt wird statt dem dpi-Wert auch direkt der *mag*-Wert im Datei- oder Verzeichnisnamen des Fonts verwendet.

Der Suchpfad für diese Datei enthält in der Regel auch noch den Namen des METAFONT-Modus, für den die Fonts erzeugt wurden, oder eine Abkürzung für diesen. Zur Kontrolle ist die Designgröße des Zeichensatzes dann auch noch in der pk-Datei selbst gespeichert. Ebenso ist der METAFONT-Modus normalerweise als Kommentar in der pk-Datei zu finden. Ob tfm- und pk-Datei zusammenpassen, wird über eine Prüfsumme kontrolliert. Diese Prüfsumme ist für alle pk-Dateien, die zu derselben tfm-Datei gehören, gleich, also insbesondere unabhängig vom METAFONT-Modus.

Findet ein DVI-Treiber eine pk-Datei nicht, so gibt es unterschiedliche Ausweichstrategien, die teilweise auch kombiniert verwendet werden.

- Es wird eine Toleranzabweichung in der Designgröße zugelassen. Diese Abweichung kann je nach DVI-Treiber bei wenigen Promille bis zu mehreren Prozent liegen und ist teilweise auch vom Anwender einstellbar. Je nach Größe der tatsächlichen Abweichung kann die Ausgabe durch den DVI-Treiber dann erkennbare oder auch für den normalen Leser nicht sichtbare Fehler aufweisen.
- Es wird ein Ersatzzeichensatz in der gewünschten Designgröße verwendet. Da dieser Zeichensatz in der Regel eine andere Fontmetrik und auch ein anderes Aussehen als der eigentlich gewünschte hat, sind die Abweichungen hier mit bloßem Auge zu erkennen und können sich auch in völlig falschen Zeichen auswirken.
- Es wird ein Ersatzzeichensatz in einer zuvor festgelegten Designgröße verwendet. Normalerweise ist dies ein Zeichensatz, der immer vorhanden sein sollte, wie etwa cmr10 in der Auflösung des Druckers. Diese Methode kann zu nichts dienen als der notdürftigen Wiedergabe des Inhaltes eines Textes.
- Statt der jeweiligen Zeichen wird entsprechender Leerraum oder ein entsprechendes Rechteck ausgegeben. Diese Methode ist häufig als letzte Lösung zu finden. Sie hat gegenüber der Methode der Ersatzzeichensätze den Vorteil, daß sie die Ausgabe weniger verfälscht, der Mißstand eindeutiger zu erkennen und der Rest besser zu beurteilen ist.
- Es wird ein Programm gestartet, das dafür sorgt, daß der fehlende Zeichensatz in der gewünschten Designgröße beispielsweise mit Hilfe von METAFONT erzeugt wird. Diese Methode ist inzwischen die Regel und teilweise soweit perfektioniert, daß bei echten PostScript-Fonts nicht METAFONT, sondern ein Programm aufgerufen wird, das pk-Fonts aus PostScript-Fonts erzeugen kann.
- In eine Datei wird eine Information über den fehlenden Zeichensatz geschrieben. Diese Datei kann dann von einem anderen Programm für die Erzeugung aller fehlenden Zeichensätze einer DVI-Datei ausgewertet werden. Diese Methode wird hauptsächlich auf Systemen mit traditionell wenig Arbeitsspeicher verwendet, bei denen fehlende Zeichensätze aus zu erwartendem Speicherplatzmangel nicht unmittelbar erzeugt werden können.
- Fast immer weist der DVI-Treiber zusätzlich zur sonstigen Vorgehensweise auf den Mißstand hin.

Gute Ergebnisse mit weniger Zeichensatzberechnungen

Eine Suche nach pk-Dateien für um den Faktor 1,414 vergrößerte Zeichensätze ergibt auf Standardsystemen, daß solche Zeichensätze normalerweise nicht existieren. Sie müssen also von METAFONT erst noch erzeugt werden. Ursache hiervon ist, daß das Vergrößerungsschema von LaTeX, das mit Befehlen wie `\small`, `\large`, `\Large` etc. verwendet wird, nicht auf $\sqrt{2}$, sondern auf 1,2 basiert. In diesem Schema wäre also $1{,}2^2 = 1{,}440$ ein möglicher Vergrößerungswert. Verwendet man diesen Wert, so lädt dvips statt `cmr10.424pk at 10.0pt` nun `cmr10.432pk at 10.0pt` in der Auflösung 300 dpi. Dieser Zeichensatz ist normalerweise direkt verfügbar. Verwendet man jedoch die T1-Codierung und EC-Fonts, so wird entsprechend `ecrm1000.432pk at 10.0pt` angefordert. Eine solche Datei ist normalerweise jedoch nicht verfügbar und muß also von METAFONT erst noch berechnet werden. Das liegt daran, daß in der OT1-Codierung tatsächlich kein cmr14 existiert und deshalb vom LaTeX-Größenumschaltbefehl stattdessen cmr10 in entsprechender Vergrößerung angefordert wird. Bei T1-Codierung existiert jedoch ein ecrm1440, also eine 14,40 pt-Variante von ecrm, die normalerweise von LaTeX auch verwendet wird. Da `\mag` jedoch kein LaTeX-Befehl ist, sondern auf TeX-Ebene durchgeführt wird, wird hier nicht eine andere Entwurfsgröße, sondern die vorgegebene Entwurfsgröße in anderer Auflösung erzeugt.

Die tolerierbar geringe Abweichung von weniger als zwei Prozent erspart also bei den OT1-codierten CM-Schriften die Berechnung einer erheblichen Menge von Zeichensätzen. Demgegenüber bringt sie bei den T1-codierten EC-Schriften keinerlei Vorteil. Zu dieser Abweichung ist noch zu bemerken, daß bei TeX und LaTeX in der Regel mit der Einheit Point (pt) gerechnet wird, wohingegen im europäischen und insbesondere deutschsprachigen Buchdruck normalerweise eigentlich Didot-Punkte (dd) verwendet werden. 1157 dd entsprechen 1238 pt. Sind also von der Druckerei eigentlich 10-Punkt-Schriften gefordert, so arbeitet man bei Verwendung von 10 pt-Schriften eigentlich mit etwas zu kleinen Zeichen, nämlich 9, 3 dd. Die Vergrößerung um den Faktor 1,440 statt 1,414 wirkt dem teilweise entgegen. Die sich dadurch ergebende Abweichung bei den Rändern liegen im Bereich eines Millimeters und ist damit in der Regel vernachlässigbar.

Fazit

Es gibt verschiedene sehr einfache Möglichkeiten, LaTeX-Dokumente zu vergrößern, beispielsweise um Druckereianforderungen zu erfüllen. Damit ist es auch

kein Problem, der Empfehlung nachzukommen, ein Dokument mit LaTeX immer in der Größe zu entwerfen, in der das Endergebnis erscheinen soll, und gegebenenfalls nur die Druckvorlagen in geänderter Größe auszugeben. Die beiden ausführlich aufgezeigten Wege der Manipulation des *mag*-Wertes sind bis auf wenige Ausnahmen äquivalent. Vernachlässigt man Fragen der Handhabung, so bleibt als tatsächliches Entscheidungskriterium lediglich die Frage, ob *true*-Größen unberührt bleiben sollen oder nicht.

Überschreiben erlaubt – das overpic-Paket

Rolf Niepraschk

Im folgenden Artikel wird das LaTeX-Paket `overpic` vorgestellt und an einem Beispiel gezeigt, wie damit EPS-Grafiken nachträglich durch LaTeX-Anweisungen ergänzt werden können.

Oft tritt der Fall ein, daß EPS-Grafiken, die in ein LaTeX-Dokument eingefügt werden sollen, noch nicht den eigenen Ansprüchen genügen. Ursache dafür können unzureichende Möglichkeiten des Mathematiksatzes im Grafikprogramm sein, oder man hat eine fertige Grafik, aber nicht das geeignete Grafikprogramm, um Änderungen durchführen zu können. Hier ist oft das Paket `PSfrag` hilfreich, um bereits vorhandene Textbestandteile der EPS-Grafik durch eigene LaTeX-Anweisungen zu ersetzen (siehe dazu [**?**] und [**?**]). Voraussetzung dafür ist, daß die Originalgrafik überhaupt Text enthält. Das im folgenden beschriebene LaTeX-Paket `overpic` bietet eine weitere Möglichkeit, um EPS-Grafiken nachträglich zu ergänzen und ist dabei nicht auf bereits vorhandene Texte in der EPS-Grafik angewiesen, erfordert andererseits jedoch die explizite Angabe der Koordinaten der Orte, an welchen die LaTeX-Anweisungen hinzugefügt werden sollen. Seine innere Struktur ist, verglichen mit `PSfrag`, sehr einfach. Das Paket `overpic` definiert eine Umgebung gleichen Namens, die eine Kombination des Befehls `\includegraphics` aus dem Paket `graphicx` und der `picture`-Umgebung darstellt. Sie lädt die als Parameter angegebene Grafik-Datei und verhält sich dann wie eine `picture`-Umgebung, die denselben Platz wie die Grafik einnimmt. Als Längeneinheit innerhalb der Umgebung wird entweder, wie in einer `picture`-Umgebung üblich, der Wert der Länge `\unitlength` oder eine relative Einheit verwendet. Innerhalb von `overpic` kann mit `\put`-Befehlen nahezu beliebiger LaTeX-Code positioniert werden (mathematische Formeln,

Literaturzitate, andere EPS-Grafiken u. v. a.). Die `overpic`-Umgebung wird folgendermaßen angewendet:

```
\usepackage[⟨Paketoptionen⟩]{overpic} % laedt graphicx und eepic
 ...
\begin{overpic}[⟨Grafikoptionen⟩]{⟨Dateiname⟩}
 ...
\end{overpic}
```

Die Parameter können folgende Werte annehmen:

⟨*Paketoptionen*⟩

- `absolute`: Absolute Positionierung. Längeneinheit ist `\unitlength`.
- `percent`: Relative Positionierung. Längeneinheit ist 1/100 der Länge der längeren Seite.
- `permil`: Relative Positionierung. Längeneinheit ist 1/1000 der Länge der längeren Seites.
- Alle anderen Paketoptionen des Paketes `graphicx`.

⟨*Grafikoptionen*⟩

- `unit=`x: Nur bei absoluter Positionierung wirksam. Statt `\unitlength` gilt in dieser `overpic`-Umgebung als Längeneinheit der mit „x“ angegebene Wert, der eine in TEX gültige Länge angeben muß.
- `grid`: Zur Orientierung wird ein bemaßtes Gitter gezeichnet.
- `tics=`x: Die Ganzzahl „x“ gibt den Abstand zwischen zwei Gitterlinien an. Standardwerte sind hierbei 10 für „`absolute`“ bzw. „`percent`“ und 100 für „`permil`“.
- Alle anderen bei `\includegraphics` erlaubten optionalen Parameter.

⟨*Dateiname*⟩

- Name der einzufügenden EPS-Datei. Sie wird in derselben Weise wie beim Befehl `\includegraphics` behandelt.

Abbildung 1 zeigt anhand eines einfachen Beispiels die Anwendung von `overpic`. Der folgende LATEX-Code wurde dafür verwendet:

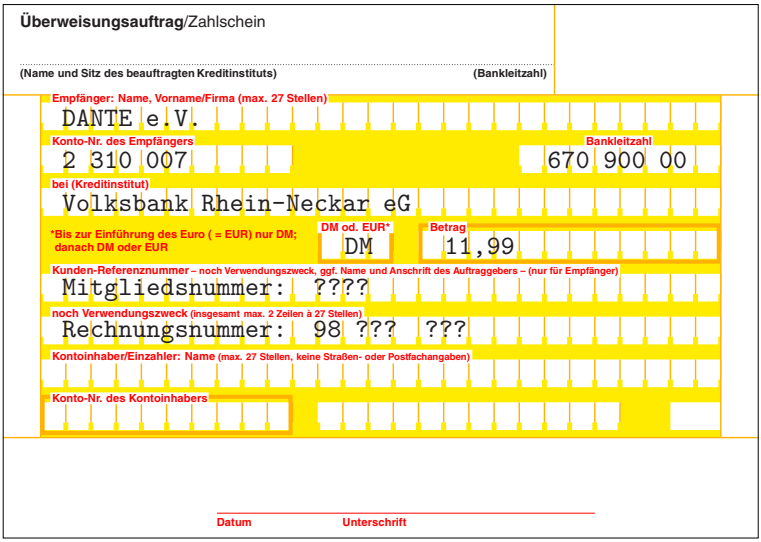

Abbildung 1: Mit `overpic` ergänzte EPS-Grafik

```
\usepackage[permil]{overpic}

\begin{figure}
  \centering
  \begin{overpic}[scale=0.66667]{ueberw.eps}
    \ttfamily \put(80,540){DANTE e.V.}
    \put(80,485){2 310 007} \put(715,485){670 900 00}
    \put(80,430){Volksbank Rhein-Neckar eG}
    \put(450,374){DM} \put(580,374){11,99}
    \put(80,319){Mitgliedsnummer: ????}
    \put(80,264){Rechnungsnummer: 98 ??? ???}
  \end{overpic}
  \caption{Mit \texttt{overpic} erg"anzte EPS-Grafik}
  \label{fig:UEBERW}
\end{figure}
```

Das `overpic`-Paket mit weiteren Beispielen ist auf dem CTAN-Server im Verzeichnis `tex-archive/macros/latex/contrib/overpic` zu finden.

Anpassung von DviWin 2.9 an Web2c für WIN32

Armin Geisse

Dieser Artikel beschreibt eine mögliche Anpassung von DviWin 2.9 an die Web2c-Implementierung für WIN32. Die Schwerpunkte liegen dabei auf der Nutzung mehrerer Druckerauflösungen sowie auf der Beschränkung der Font-Verzeichnisse.

Problematische Beziehung

Vor einiger Zeit wurde die neue CD-ROM „TEX Live 3“ an alle Mitglieder von DANTE e.V. verteilt. Diese enthält ein vollständiges TEX-System mit der Web2c-Implementierung der Programme für mehrere Plattformen, so auch für Windows 95 und Windows NT, im folgenden als WIN32 zusammengefaßt. Als DVI-Treiber wurde Windvi von Fabrice Popineau mitgeliefert. Dieser Treiber befindet sich derzeit allerdings noch im Beta-Stadium.

Als mögliche Alternative zur Nutzung von Windvi bietet sich das Programm DviWin 2.9 an – man beachte die feinen Unterschiede in der Namensgebung. Dessen Vorzüge, insbesondere die Filtertechnik, wurden von Georg Greschner in [**?**] sehr gut beschrieben. Trotz aller positiven Eigenschaften besitzt DviWin 2.9 jedoch die gravierende Einschränkung, daß das Programm nicht in der Lage ist, Verzeichnisse rekursiv nach Fontdateien zu durchsuchen. So muß jedes in Frage kommende Font-Verzeichnis explizit angegeben werden; als einzige variable Größe ist dabei die horizontale Auflösung des Fonts erlaubt. Dies erschwert die Nutzung von automatisch mit `mktexpk` generierten pk-Dateien, denn im Gegensatz zur Originalversion „verteilt“ WIN32 `mktexpk` die Dateien recht weitläufig im Font-Verzeichnisbaum. `Mktexpk` orientiert sich dabei am Verzeichnis der zugehörigen tfm-Datei. Die Aufzählung aller Font-Verzeichnisse sprengt unter diesen Umständen jeden vernünftigen Rahmen.

Es ist natürlich trotzdem möglich, automatisch generierte Fonts mit DviWin 2.9 zu nutzen, sogar für mehrere Druckerauflösungen. Dazu bedarf es allerdings einiger Anpassungen, die ich im folgenden erläutern möchte. Da auch in diesem Falle viele Wege nach Rom führen, ist meine Lösung nicht als „Königsweg“ zur Anpassung von DviWin an Web2c zu verstehen, sie ist jedoch eine Möglichkeit, die Eigenarten beider Programme aufeinander abzustimmen. Bei der Angabe von Pfaden habe ich angenommen, daß TEXLive in `c:\texlive` installiert wurde. Als Beispiel für einen Drucker fungiert der EPSON Stylus Pro mit

360 dpi und 720 dpi, eine Anpassung an andere Modelle und Auflösungen sollte jedoch problemlos durchzuführen sein.

Monogamie

Solange wir uns nur auf eine einzige Druckerauflösung und damit auch nur auf einen Modus bei der Fontgenerierung beschränken, ist `mktexpk` bei der Verteilung der Dateien relativ leicht zu bändigen. Zum Glück gibt es den Parameter `-destdir`, mit dem das Zielverzeichnis explizit vorgegeben werden kann. Um den Stylus Pro nur mit 360 dpi zu nutzen, ist in DviWin unter Options – Missing Fonts.. folgendes einzutragen:

- Execute command
  Cmd: `mktexpk -destdir c:/texlive/texmf/fonts/pk/epstypro -mfmode epstypro -bdpi 360 -mag magstep\($m) -dpi $r $f`

Dabei ist `epstypro` der in `c:\texlive\texmf\metafont\misc\modes.mf` definierte Modus für den EPSON Stylus Pro mit 360 dpi. Dieser Modus muß natürlich für andere Drucker und Auflösungen entsprechend angepaßt werden. WIN32 `mktexpk` speichert nun alle automatisch generierten pk-Dateien in `...\texmf\fonts\pk\epstypro\dpi`*xxx*, wobei *xxx* durch die jeweilige horizontale Auflösung ersetzt wird. Damit DviWin die Fonts auch findet, müssen wir in Options – Font Directory... noch den gesamten Pfad eintragen:

```
.\dpi$r;c:\texlive\texmf\fonts\pk\epstypro\dpi$r
```

Mit diesen beiden Eintragungen haben wir DviWin in die Lage versetzt, alle notwendigen Dateien zur Nutzung unseres Druckers mit 360 dpi generieren zu können.

Wenn's ein bißchen mehr sein darf

Hippocrates Sendoukas, der Autor von DviWin, hat seinem Programmpaket eine Batch-Datei beigefügt, die es über if-Abfragen ermöglicht, mit mehreren Druckermodi bzw. Auflösungen zu arbeiten. Es gibt jedoch eine (meiner Meinung nach elegantere) Lösung, die ohne Batch-Datei auskommt. Dazu definieren wir uns für jede gewünschte Druckerauflösung einen neuen Modus in `c:\texlive\texmf\metafont\misc\modes.mf`. Wichtig ist hierbei nur, daß die Namen der neuen Modi mit der jeweiligen Auflösung enden und daß sie den richtigen Druckermodi zugeordnet werden. So lautet der zusätzliche Eintrag in `modes.mf` für unser Beispiel – den Teil `dviwin` in den Bezeichnungen der neuen Modi habe ich dabei willkürlich gewählt:

```
dviwin360:=epstypro;
dviwin720:=esphi;
```

Damit `mktexpk` die neuen Modi auch erkennt, muß mit

```
mf -ini -base=mf \input plain input modes dump
```

eine neue Version der Datei `mf.base` erstellt werden, die dann in `c:\texlive\texmf\web2c` abzuspeichern ist.

Es bleibt uns jetzt nur noch, DviWin analog zum vorherigen Abschnitt anzupassen. Die Flexibilität bei der automatischen Generierung der Fonts erreichen wir durch die Abänderung unseres Cmd in:

```
mktexpk -destdir c:/texlive/texmf/fonts/pk/dviwin$X
-mfmode dviwin$X -bdpi $X -mag magstep\($m) -dpi $r $f.
```

Da sich hinter dem Parameter `$X` die jeweilige Druckerauflösung versteckt, generiert `mktexpk` automatisch die richtigen Fonts und speichert sie am korrekten Platz. Der Suchpfad für pk-Dateien wird dann noch um ein Verzeichnis ergänzt. Hier läßt sich `$X` leider nicht verwenden. Er lautet nun:

```
.\dpi$r;c:\texlive\texmf\fonts\pk\dviwin360\dpi$r; c:\texlive\
texmf\fonts\pk\dviwin720\dpi$r
```

Somit werden nun alle Fonts für 360 dpi und 720 dpi automatisch erstellt.

Die jeweiligen Einträge für einen Laserdrucker mit 300 dpi und 600 dpi könnten lauten:

```
dviwin300:=cx; dviwin600:=ljfour;
```

Im Suchpfad wäre dann folglich `360` durch `300` und `720` durch `600` zu ersetzen.

Glaubensdinge

Es ist sicher eine „Glaubensfrage“, ob der Umweg über eine Batch-Datei nicht die einfachere Lösung darstellt. Für die in diesem Artikel vorgestellte Vorgehensweise sprechen meiner Meinung nach jedoch hauptsächlich zwei Punkte:

- Änderungen und Anpassungen an die eigene Drucker-Hardware sind normalerweise auch in der von DviWin mitgelieferten Batch-Datei vorzunehmen.
- Warum soll man sich in Zeiten von Windows 95 und Windows NT immer noch für eine Batch-Datei als „Krücke“ entscheiden, wenn die (für die jeweilige Plattform optimierten) Programme direkt zu nutzen sind?

Die beschriebenen Anpassungen gelten (mit leichten Abwandlungen) auch für das Zusammenspiel der DJGPP-Version von Web2c mit DviWin unter Windows 3.x. Diesbezügliche Anfragen können direkt an mich gerichtet werden.

Rohdaten einlesen

Peter Willadt

Die Bearbeitung von Daten, die von anderen Programmen erzeugt wurden, fällt am leichtesten, wenn beim Datenexport Markup eingefügt werden kann, das TEX versteht. In diesem Artikel soll gezeigt werden, wie auch Daten, die kein Markup enthalten, jedoch in einem genau festgelegten Format vorliegen, von TEX bearbeitet werden können.

Voraussetzungen

Hierfür unterstellen wir, daß die Daten in Form von Zeilen mit festgelegter Reihenfolge vorliegen. Eine Zeile kann aus einem oder mehreren Datenfeldern bestehen. Ein Datensatz besteht aus einer bestimmten Anzahl Zeilen. Derartige Bedingungen liegen beispielsweise vor, wenn der Text-Export aus einer Datenbank vorgenommen wird. Soweit sich zusätzlicher Text einschmuggeln läßt, gelangt man zu eleganten Lösungen wie der von Ulrike Fischer [**?**] vorgestellten.

Im folgenden soll dargestellt werden, wie „roher“ Text in tabellarischer Form gesetzt werden kann, ohne den Text selbst ändern zu müssen. Hierbei sind zwei Vorgehensweisen denkbar: Ein prozeduraler Ansatz, in dem ein Steuerprogramm Zeile für Zeile einliest und bearbeitet, und ein eher *daten-getriebener*, in dem sich der Text selbst formatiert. Für beide Methoden wird ein Beispieltext verwendet, der in einer Datei namens `TESTDAT.TXT` vorliegt und wie folgt aufgebaut ist:

Name
Straße
PLZ und Ort
`Fred Ferkel`
`Ginsterweg 11`
`76123 Heumaden bei Calw`
`Gustav Gruendlich`
`Seifenstrasse 13`
`12345 Berlin`
viele weitere Zeilen

Die hier kursiv gedruckten Angaben sind in der eigentlichen Datei nicht enthalten, zudem ist jeder Datensatz vollständig. Wir unterstellen weiterhin, daß

der Text keine Zeichen enthält, die von TEX nicht bearbeitet werden können, um das Beispiel möglichst einfach zu halten.

Der daten-getriebene Ansatz

Das einzige, was mit Sicherheit in jeder Zeile vorkommt, ist das Zeilenendezeichen. TEX fügt am Zeilenende ein Zeichen mit dem Code `\endlinechar` ein, das meist auf `^^M` gesetzt ist. Es gibt jetzt die Möglichkeit, entweder dieses Zeilenendezeichen aktiv zu machen und ihm ein Makro zuzuordnen oder stattdessen per `\everypar` am Anfang jeder Zeile ein Makro auszuführen, nachdem man dem Zeilenendezeichen die Bedeutung eines Absatzendes gegeben hat. Der letzteren Methode möchte ich den Vorzug geben, da beim Einlesen der Datei mittels `\input` die erste Zeile mit bearbeitet werden soll und nach der letzten Zeile die Bearbeitung unmittelbar beendet sein sollte. Im Beispiel wird der Name und die Postleitzahl fett gesetzt, die Straße kursiv und der Ort in gewöhnlicher Schrift; Name, Straße und Ort sollen in jeweils eigenen Zeilen stehen. Die gesonderte Bearbeitung der Postleitzahl bringt mit sich, daß das Makro, das PLZ und Ortsangaben einliest, etwas komplizierter wird als die anderen.

Vor dem Einlesen der Datei und vor der Makrodefinition wird der `\catcode` von `^^M` auf `\active` gesetzt, da sonst das Zeilenende von TEX wie ein Leerzeichen behandelt wird. Es werden drei Makros namens `\name`, `\strasse` und `\ort` definiert, die die jeweiligen Zeilen formatieren und anschließend `\everypar` mit einer neuen Bedeutung belegen.

```
% Zuerst definieren...
{\catcode`\^^M=\active%
  \gdef\name#1^^M{{\bf#1}\par\everypar={\strasse}}%
  \gdef\strasse#1^^M{{\it#1}\par\everypar={\ort}}%
  \gdef\ort#1 #2^^M{{\bf#1} {\rm#2}\par\everypar={\name}}%
}
% dann anwenden:
\begingroup
  \obeylines%
  \everypar={\name}%
  \input TESTDAT.TXT
\endgroup
```

Und das war's dann auch schon. Da dem Zeilenendezeichen `^^M`, das durch die Anweisung `\obeylines` die Bedeutung eines Absatzendes zugewiesen worden

ist, von den Makros „aufgefressen“ wird, wird mittels `\par` jeweils ein Absatzende eingefügt.

Der prozedurale Ansatz

Hier verfolgen wir eine andere Strategie: Die Textdatei wird Zeile für Zeile eingelesen, jede der Zeilen wird für sich formatiert und anschließend ausgegeben. Dieser Ansatz hat seine Vorteile vor allem dann, wenn in den vorliegenden Daten Leerzeilen vorkommen können oder wenn die wechselweise Benutzung des `\everypar`-Registers als zu komplex empfunden wird. Hier kommt die Technik der *End-Rekursion* zum Einsatz. Mittels dieser wird das Makro, das einen Datensatz einliest, immerfort von sich selbst aufgerufen, bis es schließlich am Datei-Ende angelangt ist.

Bei dieser Art, Daten einzulesen, muß auch etwas Beachtung auf das Zeilenendezeichen gelenkt werden. TEX ordnet dem Zeilenende normalerweise ein Leerzeichen zu. Ist dieses nicht erwünscht, so kann es mittels `\endlinechar=-1` abgeschaltet werden. Der Teil des Makros, der die Postleitzahl setzt, ist hier noch etwas komplizierter; da die eingelesene Zeile in Form eines Makros vorliegt, muß dieses zuerst expandiert werden, bevor die Postleitzahl abgetrennt werden kann. Dies wird durch den ausgiebigen Einsatz von `\expandafter` erreicht.

```
\def\niemand{}
\def\lieseinendatensatz{
  \ifeof\testdatei     % Dateiende? Fertig.
    \let\weiter\relax
  \else                % Datensatz einlesen
    \read\testdatei to\dername
    \ifx\niemand\dername\else
      \read\testdatei to\diestrasse
      \read\testdatei to\plzort
      {\bf\dername}\par
      {\it\diestrasse}\par
      \expandafter\expandafter\expandafter%
        \splitplz\plzort xxx\par
    \fi
    % Rekursive Verknuepfung
    \let\weiter\lieseinendatensatz
  \fi
  \weiter              % Es geht weiter
```

```
}
\def\splitplz#1 #2xxx{{\bf#1} {\rm#2}}
% Datei oeffnen
\newread\testdatei
\openin\testdatei=TESTDAT.TXT
% und es geht los
{\endlinechar-1
  \lieseinendatensatz
}
\closein\testdatei
```

Im Beispiel wird zusätzlich berücksichtigt, daß manche Programme an Dateien eine überflüssige Leerzeile anhängen. Wird beim Lesen des Namens kein Text erhalten, wird deswegen der komplette Datensatz übersprungen.

Schlußbemerkung

Die Bearbeitung *roher* Daten durch TEX ist nicht ganz so einfach wie die von bereits mit Markup versehenen, aber grundsätzlich machbar. Es stehen zwei Wege zur Verfügung, die von unterschiedlichen Programmier-Paradigmen ausgehen und doch nahezu gleichwertige Ergebnisse erzielen. Dem Einsatz von TEX für die Erstellung von Serienbriefen, Katalogen und ähnlichen Listen steht damit nichts im Wege.

Was Sie schon immer über TEX wissen wollten...

Vertikal zentrierte Seiten

Bernd Raichle

Die Dokumentklasse „`book`“ sorgt mit einem impliziten `\flushbottom` dafür, daß der Textkörper auf allen Seiten gleich hoch gesetzt wird, was für den doppelseitigen Druck eine sinnvolle Annahme ist. Im Bedarfsfall kann man dies mit `\raggedbottom` in der Dokumentpräambel wieder aufheben.

Neben diesem sehr einfachen Tip kamen vor längerer Zeit Anfragen in den News-Gruppen `de.comp.text.tex` und `comp.text.tex`, die damit zusammenhängen:

> Alle Seiten, die nicht ganz voll sind, will ich vertikal auf dem Papier zentrieren. Ich kann aber keine Anweisung finden, die dies bewerkstelligt.

Leider findet man hierzu keine Lösung in den LATEX-Büchern, sondern man muß einige TEX-Kenntnisse zusammenkratzen, das TEXbook [1] vorsichtshalber aus der hintersten Ecke des Bücherregals hervorholen und sich so gewappnet in den Quellcode von LATEX wagen. Keine Angst – es ist viel einfacher, als man zuerst meint!

Ein erster Hinweis auf eine Lösung, die einem einfällt oder von anderen eingeflüstert wird, geht von `\raggedbottom` aus, da man damit schon einmal vom unteren Rand her elastischen Abstand einfügen kann. Nun muß man nur noch von oben her „zentrieren“ und da fällt der Blick vielleicht auf den mysteriösen TEX-Parameter `\topskip`. Wenn man diesen Parameter auf eine elastische statt eine starre Länge setzt, hätte man mit

```
\raggedbottom
\setlength{\topskip}{0pt plus 1fil}
```

in der Dokumentpräambel doch eine funktionierende Lösung. Oder vielleicht doch nicht?

Nein, diese Lösung ist nicht ideal, da der Parameter `\topskip` nicht für diesen Zweck existiert und auch nicht hierfür verwendet werden sollte. Der Wert von `\topskip` gibt den „Mindestabstand vom oberen Ende des Textkörpers bis zum Bezugspunkt der ersten Textzeile“ [2, S. 210] an. Er dient dazu, auf allen Seiten den Bezugspunkt der ersten Zeile des Textkörpers, unabhängig von den Ober- und Unterlängen der Lettern in dieser Zeile, an dieselbe Position zu plazieren. Daher wird `\topskip` auch auf eine starre Länge zwischen 10 pt und 30 pt gesetzt und diesen Wert sollte man nur ändern, wenn man genau weiß, was man tut! Da Gleitobjekte, wie Abbildungen und Tafeln, nicht zum Textkörper gehören, wird `\topskip` nicht vor einem Gleitobjekt eingefügt. Somit werden Gleitobjekte mit obiger „Lösung“ nicht mitzentriert.

Eine korrekte Lösung liefert ein Blick in den Kommentar und die Definitionen von `\flushbottom` und `\raggedbottom` in den Dateien `ltpage.dtx` und `ltoutput.dtx` im LaTeX 2_ε-Quellcode (bzw. für LaTeX 2.09 in `latex.tex`).

```
\newcommand{\raggedbottom}{%
  \let\@texttop=\relax \def\@textbottom{\vskip0pt plus.0001fil}}
\newcommand{\flushbottom}{%
  \let\@texttop=\relax \let\@textbottom=\relax}
```

Wie man sieht, gibt es zwei interne Makros namens `\@texttop` und `\@textbottom`, die „oberhalb“ und „unterhalb“ einer kompletten Textseite einschließlich aller Gleitobjekte expandiert und ausgeführt werden. Damit ist die Lösung für unser Problem in Form eines neuen Makros `\raggedtopandbottom` sehr einfach:

```
\newcommand{\raggedtopandbottom}{%
  \def\@textbottom{\vskip \z@ plus.0001fill}%
  \let\@texttop=\@textbottom}
```

Diese Definition muß in eine Datei mit der Namensendung `.sty` geschrieben und als Paket eingebunden werden, da die Definition LaTeX-interne Makros mit einem `@` im Namen enthält.

Zu beachten ist, daß `\newpage` und damit auch `\clearpage` und Konsorten ein `\vfil` zum „Auffüllen“ der aktuellen Seite einfügen, so daß man mit dem einfachen `fil` aus `\raggedbottom` keine zentrierten Seiten erhalten würde. Daher habe ich in der Definition von `\raggedtopandbottom` aus dem `.0001fil` ein `.0001fill` gemacht, um das `\vfil` zu „überschreiben“.

Literatur

[1] Donald E. Knuth: *The TEXbook*; Addison-Wesley Publishing Company; Reading, Mass., 1992.

[2] Leslie Lamport: *Das LATEX-Handbuch*; Addison-Wesley (Deutschland); Bonn, 1995.

Zitat des Monats

Typographie ist gelegentlich eine Kunst:
vor allem aber die Kunst von sich selbst
einmal absehen zu können.
Zur Befriedigung des Spieltriebs ist sie
ungeeignet.

Kurt Weidemann:
Wo der Buchstabe das Wort führt.
Ansichten über Schrift und Typographie.
Stuttgart 1997.

Rezensionen

„LaTeX-Tips“ von J. Kenneth Shultis

Gerd Neugebauer

Wenn ich irgendwo ein Buch aus dem TeX-Umfeld sehe, kann mich kaum etwas vom Kauf zurückhalten. Dies gilt insbesondere für solche Bücher, bei denen es sich nicht um ausgesprochene LaTeX-Einführungen handelt – ich glaube, über das Stadium bin ich schon etwas hinaus. Deshalb hat der erste Anschein mich zu einem spontanen Kauf veranlaßt.

Das Buch ist *keine* Einführung, sondern wendet sich explizit an Leser mit Grundkenntnissen. Darauf wird auch ausdrücklich im Vorwort hingewiesen. Dort sind auch einige Bücher genannt, die man gelesen haben könnte. Leider sind das vier englische Titel und von dreien habe ich noch nicht einmal etwas gehört – aber schließlich brauche ich auch keine Einführungen mehr.

Danach geht es dann auch unvermittelt ans TeXnische. Im ersten Kapitel geht es um „Schriftarten“. Erst einmal fällt auf, daß ziemlich viele Tabellen enthalten sind. Für ein Nachschlagewerk ist so etwas sicher nicht schlecht. Im Gegenzug ist aber der Text leider nicht sehr flüssig zu lesen – vielleicht hätte ich das Vorwort doch ernst nehmen sollen, in dem angedeutet ist, daß in dem Buch verschiedenste Lösungen zusammengetragen wurden. Sehr bedenklich finde ich es auch, daß in diesem Kapitel ausführlich der Befehl `\newfont` besprochen wird, ohne über die Namensnennung hinaus auf die Möglichkeiten des „New Font Selection Scheme“ einzugehen, das es erlaubt, Schriften einzubinden, die sich den LaTeX-Schriftgrößen anpassen.

Ein bißchen gestört hat mich auch, daß sehr viel mit `\def` und anderen TeX-Grundbefehlen und Plain-TeX-Makros gearbeitet wird. In einem LaTeX-Buch wäre es sicher besser gewesen, die LaTeX-Mittel zu benutzen, insbesondere wenn es keinen ersichtlichen Grund gibt, auf die TeX-Makros zurückzugreifen. Verwendung von `\newcommand` anstelle von `\def` ist einfach „politically correct“.

Im zweiten Kapitel geht es dann um „Textformatierung und Listen“. Es verdichtet sich der Verdacht, daß es sich bei dem Buch um ein Nachschlagewerk handelt. Zu den verschiedenen Themen sind jeweils ein paar Absätze geschrieben, die jemandem, der die Grundlagen mitbringt, einige neue Einsichten oder einfach Kochrezepte zu vermitteln vermögen. Hier ist mir die Übersetzung unangenehm aufgefallen. Die im Amerikanischen gebräuchlichen Anführungszeichen wurden einfach übernommen, anstatt sie den deutschen Gepflogenheiten anzupassen.

Im dritten Kapitel geht es um „Das Formatieren von Seiten“. In diesem Kapitel wird weniger eingestreuter LaTeX-Quelltext gezeigt. Das kommt der Lesbarkeit sehr zugute. Auch sind in diesem Kapitel einige Graphiken enthalten, die die verschiedenen Parameter visualisieren. So etwas habe ich schon einige Male auf Papier aufgezeichnet, deshalb hat mir das in gedruckter Form sehr gut gefallen.

Im vierten Kapitel werden „Mathematik und Gleichungen“ behandelt. Hier ist wieder sehr viel anhand von Beispielen erläutert, was der Verwendung als Kochbuch zugute kommt. Leider wird nicht auf $\mathcal{AMS}$LaTeX eingegangen, das doch einige recht nützliche Möglichkeiten für diesen Bereich bereitstellt.

Das fünfte Kapitel ist „Tabellen“ gewidmet. Anstelle der Standardlösungen für Tabellensatz wird großteils auf spezielle Makros zurückgegriffen, deren Dokumentation wohl nur kommerziell zu erhalten ist. Leider kommen in diesem Kapitel auch sehr viele Linien in den Tabellen vor – vielleicht muß ich doch irgendwann meinen Beitrag für „Die TeXnische Komödie“ über Tabellensatz fertigschreiben.

Im sechsten Kapitel geht es um „Graphik“. Leider wird ziemlich viel Raum an das Erstellen von Graphiken mit der `picture`-Umgebung und EEPIC verschwendet. Das ist nicht nur mühselig, sondern die Ergebnisse sind nur bedingt brauchbar. Zur Einbindung von fremden Graphiken, beispielsweise im PostScript-Format, werden nur Hackerlösungen angegeben. Hier wäre es in erster Linie angebracht gewesen, die Bordmittel von LaTeX einzusetzen. Analoges gilt für Bilder, die von Text umflossen werden. Anstelle der Verwendung eines der dafür vorgesehenen LaTeX-Pakete wird auf unterster Ebene mit TeX-Grundbefehlen und `\special`s gearbeitet. Alles in allem sollte man sich dieses Kapitel wohl besser ersparen.

Das siebte Kapitel ist mit „Große Dokumente“ überschrieben. Hier wird mir zum ersten Mal klar, daß das ganze Buch nur von LaTeX 2.09 handelt. Tatsächlich, kein Wort von LaTeX 2_ε im ganzen Buch. Hier noch ein Bonmot der Übersetzung: eine Überschrift lautet „Kreuzreferenzen“. Das war wohl „crossreferences“ und sollte besser mit „Querverweise“ übersetzt werden.

Das achte Kapitel behandelt „Nützliche Stile“. Einige Erweiterungen werden vorgestellt. Inzwischen sind aber alle vorgestellten Stile als Pakete für LaTeX 2_ε vorhanden. Trotzdem hat mich dieses Kapitel angeregt, etwas Neues auszuprobieren. Auch wenn man ein Paket schon mehrmals benutzt hat, kann man in uralten Beschreibungen noch etwas Neues entdecken. Wenn man den LaTeX-Begleiter schon im Schrank hat, kann man sich dieses Kapitel auch getrost ersparen.

Im anschließenden neunten Kapitel werden „Makros und verschiedene Tricks“ vorgestellt. Hier geht es zum Teil auch wieder in die Innereien von TeX. Die vorgestellten Lösungen stehen meistens ohne tiefere Erklärungen. Damit sind sie als Kochrezepte oder als Anregung brauchbar, die man zum Knobeln nutzen kann, um die dahinterliegenden Ideen zu verstehen.

Alles in allem ist das Buch etwas hinter der Zeit zurück. Trotzdem kann es denjenigen als Anregung dienen, die beispielsweise auch in den News-Gruppen aktiv sind. Aus dieser Quelle stammt auch wohl ein Teil des Inhalts. Wenn man also gerade an irgendeinem Strand liegt und keinen Internet-Zugang in greifbarer Nähe hat, kann man so den Entzugserscheinungen entgegenwirken und ein nostalgisches Gefühl heraufbeschwören: „Ach ja, so war das damals, in der guten alten Zeit“.

J. Kenneth Shultis
LaTeX-Tips – Praktische Hinweise zur Erstellung technischer Dokumente
Prentice Hall Verlag GmbH, München 1995
xiii+194 Seiten
ISBN 3-930436-25-6, DM 39,80

Leserbriefe

Addendum

Dr. Georg S. Greschner

Der in meinem Aufsatz [**?**] in Fußnote 8 auf Seite 24 erwähnte PostScript-Tintenstrahldrucker Lexmark Optra Color 40, der in meinen Tests so gut abgeschnitten hatte, kostet zur Zeit mit 4 MB internem Speicher DM 1 090 ohne MWSt. (Listenpreis). Die Firma Lexmark besteht glücklicherweise nicht darauf, daß man die benötigten weiteren 64 MB Speicher bei ihr als Modul kauft. Ich installierte statt dessen ein preiswertes Markenmodul SIMM EDO 64 MB/ 60 ns/72 pin ohne *parity check*, bestückt mit 8 Siemens-Chips von je 8 MB, das für DM 175 inkl. MWSt. im Großhandel zu haben ist, und der Drucker lief genauso gut wie mit dem sehr teuren Lexmark-Originalspeichermodul 64 MB Toshiba/50ns.

Georg S. Greschner

Spielplan

Termine

24.2.–26.2.1999 DANTE'99 und
20. Mitgliederversammlung von DANTE e.V.
Universität Dortmund
Kontakt: Stephan Lehmke

26.4.–30.4.1999 XML Europe'99
Palacio de Exposiciones y Congresos
Granada, Spanien
Kontakt: Pamela Gennusa

1.5.–3.5.1999 BachoTEX'99
Bachotek, Brodnica Lake District, Polen
Kontakt: Jola Szelatyñska

11.5.–14.5.1999 8th International World Wide Web Conference
Metro Toronto Convention Centre
Toronto, Ontario
Kontakt: WWW8, Foretec Seminars

18.5.–20.5.1999 GUT'99
Lyon, Frankreich
Kontakt: GUTenberg

15.8.–19.8.1999 TUG'99 – 20th annual meeting of the TEX Users Group
University of British Columbia
Vancouver, British Columbia
Kontakt: TEX Users Group

20.9.–23.9.1999 EuroTEX'99 – 11th European TEX Conference
Universität Heidelberg
Kontakt: Joachim Lammarsch

13.10.–18.10.1999 51. Frankfurter Buchmesse
Frankfurt
Kontakt: Messe Franfurt

Stammtische

In verschiedenen Städten im Einzugsbereich von DANTE e.V. finden regelmäßig Treffen von TEX-Anwendern statt, die für Jeden offen sind. Im WWW gibt es aktuelle Informationen unter `http://www.dante.de/dante/Stammtische.html`.

Berlin – Rolf Niepraschk
Tel.: 0 30/3 48 13 16
`niepraschk@ptb.de`
Gaststätte „Bärenschenke“
Friedrichstr. 124
Zweiter Donnerstag im Monat, 19.00 Uhr

Bremen – Martin Schröder
Tel.: 04 21/2 23 94 25
`ms@dream.hb.north.de`
Universität Bremen, Unikum
Erster Donnerstag im Monat, 18.00 Uhr

Dortmund – Stephan Lehmke
`Stephan.Lehmke@cs.uni-dortmund.de`
Cafe Durchblick
Universität Dortmund, Campus Nord
Zweiter Mittwoch im Monat, 20.00 Uhr

Dresden – Torsten Schütze
Tel.: 03 51/463-40 84
`schuetze@math.tu-dresden.de`
Klub Neue Mensa
Letzter Mittwoch im Monat, 19.00 Uhr

Freiburg – Heiko Oberdiek
Tel.: 07 61/4 34 05
`oberdiek@ruf.uni-freiburg.de`
Gaststätte „Aguila“
Sautierstr. 19
Dritter Donnerstag im Monat, 19.30 Uhr

Hamburg – Volker Hüttenrauch
`volker_huettenrauch@hh.maus.de`
Letzter Donnerstag im Monat, 18.00 Uhr

Hannover – Stephanie Hinrichs
Regionales Rechenzentrum
Schloßwender Str. 5
Tel.: 05 11/7 62 43 82
`hinrichs@rrzn.uni-hannover.de`
Seminarraum RRZN
Zweiter Mittwoch von geraden Monaten, 18.30 Uhr

Heidelberg – Luzia Dietsche
Tel.: 0 62 21/54 45 27
`luzia.dietsche@urz.uni-heidelberg.de`
China-Restaurant Palast
Lessingstr. 36
Letzter Mittwoch im Monat, 20.00 Uhr

Karlsruhe – Klaus Braune
Tel.: 07 21/6 08 40 31
`braune@rz.uni-karlsruhe.de`
Universität Karlsruhe, Rechenzentrum
Zirkel 2, 3. OG Raum 316
Erster Donnerstag im Monat, 19.30 Uhr

Köln – Daniel Schlieper
`tex-ws@rrz.uni-koeln.de`
Zentrum für Paralleles Rechnen,
Weyertal 80
Vierter Dienstag im Monat, 20.00 Uhr

Stuttgart – Marcus Schweizer
Tel.: 07 11/6 85 44 44
`schweiz@theochem.uni-stuttgart.de`
Wechselnd
Zweiter Dienstag im Monat, 19.30 Uhr

Wiesbaden – Christian Kayssner
Tel.: 06 11/4 81 17
Andreas Klause, Elsässer Platz 3
Erster Montag im Monat, 20.00 Uhr

Wuppertal – Andreas Schrell
Tel.: 02 02/50 63 81
`schrell@wupperonline.de`
Croatia „Haus Johannisberg“, Südstr. 10,
an der Schwimmoper Wuppertal-Elberfeld
Zweiter Donnerstag im Monat, 19.30 Uhr

Adressen

DANTE, Deutschsprachige Anwendervereinigung TEX e.V.
Postfach 10 18 40
69008 Heidelberg

Tel.:	0 62 21/2 97 66 (Mo–Fr, 10^{00}–12^{00} Uhr)	
Fax:	0 62 21/16 79 06	
E-Mail:	`dante@dante.de`	
Konten:	Volksbank Rhein-Neckar eG BLZ 670 900 00, Kontonummer 2 310 007	*neu*
	Postbank Karlsruhe BLZ 660 100 75, Kontonummer 1990 66-752	nur für Tagungen
Beiträge:	ermäßigte Mitgliedschaft	60,– DM
	Privatmitgliedschaft	80,– DM
	Institutionen des öffentlichen Rechts und Forschungseinrichtungen	120,– DM
	Firmen, die TEX anwenden	300,– DM
	Firmen, die Produkte in Verbindung mit TEX anbieten	500,– DM

Präsidium

Präsident:	Thomas Koch	`president@dante.de`
Vizepräsident:	Volker RW Schaa	`vice-president@dante.de`
Schatzmeister:	Horst Szillat	`treasurer@dante.de`
Schriftführer:	Günter Partosch	`secretary@dante.de`
Beisitzer:	Arnulf Liebing	`adviser@dante.de`

Server

ftp:	`ftp.dante.de` [129.206.100.192]
E-Mail:	`ftpmail@dante.de`
WWW:	`http://www.dante.de/`
Mailbox:	0 62 21/16 84 26

Die TEXnische Komödie

11. Jahrgang Heft 1/1999 Februar 1999