DANTE

Deutschsprachige Anwendervereinigung TEX e.V.

Die TEXnische Komödie

Heft 4/1994 6. Jahrgang Februar 1995

Impressum

Beiträge sollten in Standard-LATEX-Quellcode an untenstehende Anschrift geschickt werden (entweder per e-mail oder auf Diskette). Sind spezielle Makros oder Stylefiles dafür nötig, so müssen auch diese mitgeliefert werden. Außerdem müssen sie auf Anfrage Interessierten zugänglich gemacht werden.

Diese Ausgabe wurde mit Hilfe von `PubliC TeX--XeT, Version 3.141--1.1/DOS-TP 1.4`, LATEX `2.09` v. 25.3.1992, `dviscr 1.4s` (für die Bildschirmdarstellung), `dvihplj 1.4s` (für die Korrektur) und `dvipsk 5.55a` (für die endgültige Belichtung) fertiggestellt.

Erscheinungsweise: vierteljährlich
Erscheinungsort: Heidelberg
Auflage: 3000

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

Tel.: 06221/2 97 66
Fax: 06221/16 79 06
e-mail: `dante@dante.de`

Belichtung: LaserSatz Thewalt
Kapellenweg 8
69257 Wiesenbach
Tel.: 06223/4 83 14

Druck: VOD Vereinigte Offsetdruckereien Mannheim Heidelberg
Handelsstr. 13
69214 Eppelheim

Redaktionsschluß für Heft 1/1995: 31.3.1995

Editorial

Liebe Leserinnen und Leser,

da diese Ausgabe der Mitgliederzeitung im Gegensatz zu den letzten sehr viel dünner ausfällt, werde ich mich dem im Editorial anpassen und nicht viele Worte machen. Sie ist zwar immer noch nicht „in time“, läßt aber zumindest nicht so lange auf sich warten wie die vergangenen ...

Nur auf eines möchte ich hinweisen: es hat sich endlich jemand gefunden, der bereit ist, eine Artikelserie zu veröffentlichen. Sie dürfen also ab jetzt damit rechnen, etwas über „Orale Spielereien mit TEX“ zu erfahren. Der Autor wünscht sich, damit viele Leser zur Auseinandersetzung mit den „(Un-)Tiefen des TEXbook“ anregen zu können. Ich wünsche mir, daß der eine oder die andere die Idee einer Artikelserie aufgreift und der Redaktion ein ähnliches Angebot machen möge ...

Ihre Luzia Dietsche

Bretter, die die Welt bedeuten

BibTool – Manipulation von BibTeX-Dateien

Gerd Neugebauer

Zusammenfassung

Es war einmal, da verspürte ich das dringende Bedürfnis, BibTeX-Dateien aus verschiedenen Quellen zusammen weiterzuverwenden. Da es sich dabei nicht nur um einige wenige Einträge in den BibTeX-Dateien handelte, ergaben sich daraus diverse Aufgaben, die gelöst werden mußten. Das einfachste Problem war, daß das Erscheinungsbild der Dateien vereinheitlicht werden sollte. Das bedeutet, daß die Dateien neu formatiert werden müssen. Das nächste Problem war die Vereinheitlichung der Schlüssel, unter denen die BibTeX-Einträge angesprochen werden. Schließlich und endlich ergab sich das Problem, die Dateien zu sortieren.

Da es damals keine entsprechenden Werkzeuge gab – insbesondere nicht für den Rechner, den ich damals verwendete – wurde das BibTool-Programm geboren. Als dann einmal die grundlegenden Routinen geschrieben waren, kamen noch weitere Funktionen hinzu, sodaß man BibTool heute als das Schweizer-Armee-Messer zur BibTeX-Vorverarbeitung bezeichnen könnte.

BibTeX – Eine Würdigung

An Anfang aller Bemühungen steht BibTeX[1] [1, 2, 3]. BibTeX wird meistens als Programm bezeichnet, das Literaturzitate mit einem TeX-Dokument kombiniert. In Wirklichkeit ist BibTeX viel allgemeiner. BibTeX bietet eine einfache „Datenbank“ mit der Möglichkeit, Datensätze daraus in ein TeX-Dokument zu übernehmen. Dabei sind einige Manipulationen möglich, die hauptsächlich von der Anwendung als Literaturdatenbank inspiriert sind.

Um die Konzepte von BibTeX aufzuzeigen, betrachten wir kurz ein Beispiel. Als erstes brauchen wir eine BibTeX-Datei, die die Literaturzitate enthält. Wir betrachten den folgenden Datensatz aus der BibTeX-Datei `lit.bib`.

[1] Zur Zeit ist BibTeX 0.99c aktuell. Der Aufbau der Literaturdatenbankdateien für die schon lange angedrohte Version 1.0 sollte sich aber nicht allzu sehr unterscheiden – sollte sie jemals veröffentlicht werden.

```
@Book{gms:companion,
  author = "Michel Goossens and Frank Mittelbach and Alexander Samarin",
  title = "The {\LaTeX} Companion",
  publisher = "Addison-Wesley",
  year = 1994
}
```

Um nun ein Zitat dieser Literaturstelle in unser LaTeX-Dokument aufzunehmen, geben wir dort den Schlüssel in einem Befehl der Form `\cite{gms:companion}` an. Weiterhin müssen wir festlegen, welchen Stil wir für die Referenzen bevorzugen. Dies geschieht z. B. durch die Anweisung `\bibliographystyle{alpha}`. Schließlich muß noch angegeben werden, welchen Dateien die Referenzen zu entnehmen sind. Dies geschieht mit dem Befehl `\bibliography{lit}`. Dieser Befehl legt auch fest, an welcher Stelle die Literaturliste eingefügt werden soll.

Damit haben wir die Vorbereitungen in unserem Dokument abgeschlossen. Jetzt wollen wir uns das Ergebnis unserer Bemühungen ansehen. Als erstes wird LaTeX aufgerufen. Damit werden die Informationen aus den Befehlen `\cite`, `\bibliographystyle` und `\bibliography` in die `.aux`-Datei übernommen. Nun rufen wir BibTeX auf. BibTeX liest die `.aux`-Datei und erzeugt die `.bbl`-Datei. Ein weiterer LaTeX-Lauf befördert die Literaturliste in die `.dvi`-Datei und die Zitat-Schlüssel in die neue `.aux`-Datei, von wo aus sie durch einen anschließenden LaTeX-Lauf in die Ausgabe übernommen werden.

Soweit so gut. Einige Dinge mußten bisher beachtet werden. In der BibTeX-Datei sind Fehler verheerend. BibTeX hat eine sehr grobe Methode der Fehlerbehandlung: es ignoriert einfach alles, was noch in dem aktuellen Datensatz stehen mag. Dies geschieht insbesondere dann, wenn die Schlüssel, in unserem Beispiel ist das `gms:companion`, nicht für alle Datensätze verschieden sind. Eindeutige Schlüssel sind insbesondere dann schwierig zu garantieren, wenn man BibTeX-Dateien anderer mitbenutzen will. Hier setzt BibTool an.

BibTool – **Analyse und Neuformatierung**

Die erste Aufgabe, die anfällt, wenn man eine fremde BibTeX-Datei in die Hände bekommt, ist es, diese auf Fehler zu untersuchen und möglicherweise auch in eine „schönere“ Form zu bringen. Das ist die einfachste Aufgabe für BibTool. Ein Aufruf der Form

`bibtool` *infile* `-o` *outfile*

analysiert die Datei *infile* und legt das Ergebnis neuformatiert in der Datei *outfile* ab. Dabei gehen wir, wie im Rest dieser Beschreibung, von einem Aufruf über einen Kommando-Interpreter aus.

Das Format läßt sich dabei von einer Vielzahl von Parametern bestimmen. Diese Parameter sind mit „sinnvollen“ Vorgaben versehen. Danach sieht unser Beispiel folgendermaßen aus:

```
@Book{            gms:companion,
  author        = "Michel Goossens and Frank Mittelbach and Alexander
                  Samarin",
  title         = "The {\LaTeX} Companion",
  publisher     = "Addison-Wesley",
  year          = 1994
}
```

Auf jeden Fall werden wir aber bei diesem Durchgang auch auf Fehler aufmerksam gemacht. In einigen Fällen wird versucht, diese Fehler zu korrigieren. So werden zum Beispiel fehlende oder überzählige Kommata zwischen den Feldern ergänzt bzw. gelöscht.

BibTool – **Sortieren**

Als nächstes wollen wir eine BibTeX-Datei sortieren. Das ist auch bei unseren eigenen Dateien gelegentlich ganz nützlich. Dies geschieht mit dem folgenden Kommando:

`bibtool -s` *infile* `-o` *outfile*

Die Ausgabe-Datei *outfile* enthält die Eingabe-Datei *infile*, wobei die Datensätze entsprechend den Schlüsseln sortiert erscheinen. Diese Sortierreihenfolge – nach den Schlüsseln – ist nicht immer wünschenswert. Deshalb kann es sich als sinnvoll erweisen, vor dem Sortieren neue Schlüssel zu erzeugen. Darauf wird im nächsten Abschnitt näher eingegangen. Alternativ kann mit denselben Mitteln, die es erlauben, einen Schlüssel zu generieren, auch ein Sortierschlüssel spezifiziert werden.

BibTool – **Schlüssel**

Ein großes Problem beim Umgang mit BibTeX stellen die Schlüssel dar. Als Schlüssel könnte, mit Ausnahme einiger Zeichen, jeder beliebiger Text verwendet werden. Üblicherweise wird man jedoch seine Schlüssel so wählen, daß sie

einfach zu merken sind. Würde man einfach sinnlose Wörter als Schlüssel verwenden, müßte man sonst bei jedem Zitat den Schlüssel aus der BibTeX-Datei heraussuchen. Folglich bietet es sich an, die Schlüssel nach einem einheitlichen Verfahren zu generieren. Damit kann man einfach anhand der gewünschten Literaturstelle den zugehörigen Schlüssel ableiten und muß ihn nicht nachsehen. Dadurch entfällt ebenfalls die Aufgabe, sich für jede Literaturstelle einen neuen Schlüssel überlegen zu müssen.

BibTool enthält einen mächtigen Apparat, der es erlaubt, eindeutige Schlüssel für einen Datensatz zu generieren. Dieser Apparat ist sehr flexibel und in einem hohen Grade konfigurierbar. Im einfachsten Fall jedoch kann eine (sinnvolle) Vorgabe benutzt werden. Dies geschieht mit dem Aufruf

`bibtool -k` *infile* `-o` *outfile*

Der Parameter `-k` bewirkt die Generierung von neuen Schlüsseln. Unser Beispiel-Datensatz würde danach den Schlüssel `goossens.mittelbach.ea:companion` erhalten. Dieser setzt sich aus den Namen der Autoren (höchstens zwei werden explizit gemacht)[2] und dem ersten „relevanten“ Wort zusammen. Die Anzahl der expliziten Autoren und der relevanten Worte sind natürlich konfigurierbar. Die Worte, die nicht als relevant angesehen werden sollen, sind in einer Liste enthalten, die beliebig erweitert werden kann.

Diese Art der Schlüsselgenerierung spiegelt eine sehr persönliche Vorliebe des Autors wieder. Deshalb ist es in BibTool möglich, einen Format-String für die Generierung der Schlüssel anzugeben. Diese Bezeichnung ist eigentlich reichlich untertrieben, da die Möglichkeiten fast zu einer kleinen Programmiersprache ausgeweitet wurden.

Um einen Eindruck von den Möglichkeiten zu geben, betrachten wir einige Beispiele. Am Anfang war der Versuch, die Format-Strings der Programmiersprache C zu imitieren. Als erstes werden alle Zeichen, die keine besondere Bedeutung haben, identisch in den zu generierenden Schlüssel übernommen. Da das `%` kein erlaubtes Zeichen für einen Schlüssel darstellt, kann es zur Einleitung einer Format-Angabe benutzt werden. Das sieht etwa so aus: `%s`. Damit werden alle erlaubten Zeichen in den Schlüssel übernommen. Um anzugeben, welches Feld gemeint ist, wird dieses einfach in Klammern dahinter angegeben: `%s(title)`. Damit haben wir einen gültigen Format-String.

Wie in C ist es auch in BibTool möglich, Parameter zwischen dem `%` und dem Format-Buchstaben – hier `s` – einzufügen. In diesem Fall kann es eine Zahl sein, die angibt, wieviele Zeichen maximal übernommen werden sollen. Also ergibt

[2] `ea` steht für *et alii* (und andere).

`%16s(title)` die ersten erlaubten Zeichen des Feldes `title`, aber höchstens 16 solche Zeichen.

Andere Format-Angaben erlauben die Formatierung eines Namens (`%n`), eines Titels (`%T`) – es werden nur relevante Worte übernommen – und einer Zahl (`%d`) – es werden nur Ziffern übernommen.

Bisher wurde nicht gesagt, was passiert, wenn das angegebene Feld nicht in dem Datensatz enthalten ist. In diesem Fall scheitert hier der Versuch, einen Schlüssel zu generieren. Falls eine Alternative gegeben wurde, wird die nächste Alternative versucht. Alternative Teile eines Schlüssels lassen sich durch das Zeichen `#` trennen und in geschweifte Klammern `{ }` einschließen.

Betrachten wir den folgenden Format-String:

```
{ %n(author) # %n(editor) # nobody }
```

Er spezifiziert drei Alternativen. Diese sind durch `#` getrennt. Zuerst wird versucht, das Feld `author` zu benutzen. Ist es nicht vorhanden, dann wird das Feld `editor` versucht. War dies auch erfolglos, so wird der konstante Wert `nobody` als Schlüssel eingetragen.

An diesem Beispiel sehen wir ebenfalls, daß Leerzeichen (fast) beliebig eingestreut werden können. Da Leerzeichen nicht in einem Schlüssel vorkommen können, werden sie nicht in den automatisch erzeugten Schlüssel übernommen. Sie können jedoch die Lesbarkeit eines Format-Strings erhöhen.

Zusätzlich zu den Alternativen gibt es noch ein *if-then-else*-Konstrukt, bei dem explizit angegeben werden kann, was im Falle der Existenz oder Nichtexistenz eines Feldes zu geschehen hat.

Ein Punkt sei noch zum Abschluß zu erwähnen. Dies betrifft die Formatierung von Namen und Titeln. Hier kann es geschehen, daß die entsprechenden Felder TeX-Makros enthalten. Deshalb ist in BibTool ein kleines Modul enthalten, das TeX-Makros expandiert. Dies kann z. B. dazu benutzt werden, die vordefinierten Makros, wie `\TeX` oder `\LaTeX` in die entsprechenden Strings umzuwandeln – ansonsten werden alle unbekannten Makros einfach zum leeren String expandiert. Weiterhin ergibt sich hierdurch die Möglichkeit, die Umlaute entsprechend umzuwandeln.

Ohne vordefinierte Makros wird z. B. aus dem Autor `M"uller` der Teil eines Schlüssels `mller`. Werden die entsprechenden Makros aktiviert, so ergibt sich `mueller`, was sicher die bessere Alternative darstellt.

BibTool – **Konfigurieren**

Wir haben von der Konfigurierbarkeit von BibTool gehört. Diese kann über die Kommandozeile erfolgen, was aber ziemlich umständlich ist, insbesondere, wenn wir an längere Spezifikationen von Schlüsseln denken. Deshalb kann die Konfiguration in einer Konfigurationsdatei, der sogenannten Resource-Datei, abgelegt werden. Diese Resource-Dateien werden entweder automatisch oder durch einen speziellen Befehl von BibTool ausgewertet.

In den Quellen von BibTool sind einige nützliche Beispiele für Resource-Dateien enthalten, die entweder für eigene Entwicklungen hinzugeladen oder modifiziert werden können. Darunter ist z. B. die bereits erwähnte Definition von wichtigen TeX-Makros.

BibTool – **Extrahieren**

Nun haben wir also erfolgreich megabyte-weise BibTeX-Dateien zusammengesammelt und benutzen diese fleißig. Aber nun wollen wir ein Dokument zusammen mit den Referenzen in Form einer BibTeX-Datei an jemanden weitergeben. Also müssen wir die Referenzen heraussuchen, die in diesem Dokument benutzt werden, und sie in eine eigene BibTeX-Datei schreiben. Aber halt – wir haben ja noch BibTool. BibTool kann uns nämlich diese Aufgabe abnehmen.

Dazu müssen wir einfach die entsprechende `.aux`-Datei erzeugen. Wir haben im ersten Abschnitt gesehen, daß diese Datei die Information enthält, welche Zitate in einem Dokument vorkommen. Weiterhin ist dort die Information enthalten, welche BibTeX-Dateien benutzt werden sollen. Diese Informationen kann BibTool auswerten. Durch einen Aufruf der Form

`bibtool -x` *auxfile* `-o` *outfile*

werden die benutzten Datensätze in die Ausgabedatei *outfile* geschrieben, die genügen sollte, um alle Referenzen des ursprünglichen Dokumentes aufzulösen.

Weitere Möglichkeiten

Die Möglichkeiten von BibTool sind mit den beschriebenen Anwendungen noch nicht ausgeschöpft. Weitere Möglichkeiten sollen hier nur noch stichwortartig vorgestellt werden:

Hinzufügen und Löschen von Feldern kann z. B. dazu benutzt werden, um automatisch die Herkunft zu markieren oder eine Zeitmarke anzubringen.

Extraktion anhand von regulären Ausdrücken erlaubt eine weitere Zerlegung von BibTeX-Dateien, z. B. anhand des ersten Buchstabens des Schlüssels.

Feld-Umformung ermöglicht das konsistente Ändern aller Felder anhand von vorzugebenden Regeln.

Semantische Überprüfungen ermöglichen es, über die syntaktischen Eigenschaften hinaus Überprüfungen durchführen zu lassen, die die Plausibilität der Werte überprüfen. Das kann beispielsweise ein Test sein, ob das `year`-Feld einen sinnvollen Wert enthält.

Expansion von BibTeX-*Strings* nimmt den Mechanismus von BibTeX vorweg, Makros in Strings umzuwandeln.

Statistiken über die analysierten BibTeX-Dateien können ausgegeben werden.

Bisher unerwähnt blieb auch noch, daß die vorgestellten Möglichkeiten nicht nur einzeln genutzt, sondern (fast) beliebig kombiniert werden können.

Quellen

BibTool ist in C geschrieben. Jeder ANSI-C-Compiler sollte in der Lage sein, ein ausführbares Programm zu erzeugen. Bisher ist es zumindest schon auf UNIX-Workstations (Sun, HP, Linux), Atari und unter MS-DOS erfolgreich eingesetzt worden. BibTool wird in Source-Form verteilt. Binär-Versionen sind beim Autoren *nicht* erhältlich.

Wer nach dieser Kurzbeschreibung Geschmack auf BibTool bekommen hat, kann es sich aus verschiedenen Quellen besorgen. An erster Stelle sei hier natürlich der Server von DANTE e.V. (`ftp.dante.de`) genannt. Dort ist BibTool in

```
tex-archive/biblio/bibtex/utils/bibtool/BibTool
```

Selbstverständlich ist es damit auch auf allen CD-ROM-Abzügen der CTAN-Server (z. B. PrimeTimeFreeware TeXcetera oder die CD von DANTE e.V. und Addison-Wesley) zu finden.

Literatur

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

[2] Leslie Lamport: LaTeX: A Document Preparation System, Addison-Wesley, Reading, Mass., 2. Auflage, 1994.

[3] Oren Patashnik: BibTeXing, 1988.

Anmerkung der Redaktion: `bibclean`, ein mit BibTool vergleichbares Werkzeug, stammt aus der Feder Nelson Beebes und ist in dessen Artikel „Bibliography Prettyprinting and Syntax Checking“ beschrieben, der in TUGboat 14(1993), No. 4, S. 395–419 veröffentlicht wurde. Dieses Werkzeug kann man auch auf dem CTAN-Server von DANTE e.V. (`ftp.dante.de`) im Verzeichnis

```
tex-archive/biblio/bibtex/utils/bibclean/
```

als C-Quelle finden. Im Gegensatz zu BibTool erledigt `bibclean` nur den „Syntaxcheck“ und das „Prettyprinting“ von Datenbankdateien, wobei man es durch Konfigurationsdateien an seinen eigenen Stil anpassen kann. Als weitere Option kann es als Frontend benutzt werden, das eine Datenbankdatei in lexikalische Tokens wandelt, die ein nachgeschalteter Parser weiterverarbeiten kann. Neben der Beschreibung von `bibclean` stellt Nelson Beebe eine Analyse des Formats der Literaturdatenbankdateien vor, die BibTeX verwendet. Aus dieser Analyse entwickelt er eine Grammatik, die auch in diesem Artikel veröffentlich ist, und die man für eigene Programmentwicklungen, die diese Datenbankdateien verwenden, nutzen sollte.

Bernd Raichle

Computer Modern Bright

Walter Schmidt

Die Endstrichlose oder Grotesk ist eine
zur Zeit fast unerträglich häufige Schriftart.
Es wäre gut, wenn man ihren Gebrauch auf Schlagzeilen
und wichtige öffentliche Aufschriften ... beschränkte.

J. Tschichold, 1960

Einst als „grotesk“ verschrien, sind die serifenlosen Schriften heute auch aus der anspruchsvollen Typographie nicht mehr wegzudenken. Wer als Anwender(in) von TeX oder LaTeX PostScript einsetzt, kann an dieser Entwicklung teilhaben; ist man jedoch auf die Computer-Modern-Fonts angewiesen, dann steht

als einzige serifenlose Textschrift die CM Sans Serif zur Auswahl. Sie eignet sich hervorragend, um innerhalb eines mit CM Roman geschriebenen Textes Wörter oder Textteile hervorzuheben, jedoch weniger für normalen Fließtext. Aus diesem Grund habe ich versucht, aufbauend auf dem Konzept der Computer-Modern-Fonts, eine alternative Groteskschrift zu gestalten.

Der Entwurf

Alle Computer-Modern-Schriften gehen auf ein einheitliches in METAFONT codiertes Design zurück, das durch 62 Parameter in ganz unterschiedliche äußere Formen gebracht werden kann. Dieses Konzept und die Bedeutung der Parameter sind in [1] beschrieben. Allein durch Variation der Parameter und, im Idealfall, ohne METAFONT-Programmierung ist es also möglich, neue Schriften zu erzeugen.

Manche der Parameter hängen in höchst komplizierter Weise in ihrer Wirkung voneinander ab, so daß es sich empfiehlt, von den vorhandenen Schriften auszugehen und Änderungen zunächst nur äußerst zurückhaltend vorzunehmen. Eine Alternative besteht darin, komplette Gruppen von zueinander passenden Parametern aus den Standardschriften zu übernehmen.

Mein Entwurfsziel war eine serifenlose Schrift, die sich im Vergleich zur CM Sans durch höhere Mittellängen und deutlich geringere Strichstärken auszeichnen sollte („light“). Sie sollte kein Ersatz für die CM Sans sein und auch nicht unbedingt mit der CM Roman harmonieren. Betrachtet man die Eigenschaften der vorhandenen CM-Schriften, dann erkennt man, wie sich die Parameter dafür zumindest ansatzweise aus vorhandenen Schriften zusammenstellen lassen:

- Die Breite der Zeichen wird aus der `cmss10` übernommen.
- Mit der `cmssdc10` existiert bereits eine Schrift mit den gewünschten vertikalen Proportionen, vor allem den höheren Mittellängen.
- `cmssq8` liefert die Parameter für die Strichstärken, Radien usw; diese Schrift wirkt erheblich leichter als `cmss10`.

Das Ergebnis aus dieser Kombination bedurfte allerdings noch einiger offensichtlicher Korrekturen, von denen ich zwei etwas schwierige näher erläutern will, und zwar die Festlegung der Laufweite und das Design der punktförmigen Akzente:

In den CM-Fonts werden die Abstände zwischen den Zeichen unter anderem über drei Parameter (`letter_fit`, `serif_fit`, `cap_serif_fit`) gesteuert. Mit den aus `cmss10` oder `cmssq8` übernommenen Parametertripeln ergaben sich aber jeweils etwas ungleichmäßige Zeichenabstände. Mangels Kenntnis einer systematischen Lösung blieb mir nichts anderes übrig, als versuchsweise kleine Änderungen an diesen Vorgaben anzubringen und anhand des Schriftbildes in Originalgröße zu beurteilen. Das so erreichte Ergebnis ist sicher noch nicht optimal. Vielleicht wird es sogar nötig sein, Korrekturen nicht nur an den Parametern, sondern auch an den Kerning-Tabellen oder gar am Entwurf der Einzelzeichen vorzunehmen.

Das Konstruktionsprinzip der CM-Fonts garantiert nicht, daß der Durchmesser der Punkte auf den Buchstaben i und j exakt so breit ausfällt wie der Grundstrich. Zumindest für perfekt eckige i-Punkte wollte ich das aber voraussetzen. Abhilfe war hier nur durch einen Eingriff in die METAFONT-Programme möglich: Im i, j und den Ligaturen f-i und f-f-i wird die Größe der Punkte jetzt aus dem Grundstrich des i abgeleitet. Eine weitere Änderung war bei den Pünktchen auf den Umlauten notwendig, die in ihrer Größe mit den i-Punkten harmonieren sollen, aber – z. B. beim fetten Schriftschnitt – auch nicht zu groß werden dürfen.

Im Zusammenhang mit den Akzenten trat schließlich noch das Problem auf, daß ihre Höhe über dem Grundsymbol u. a. aus der x-Höhe berechnet wird. Der Algorithmus ist für die Standardschriften mit ihren niedrigen Mittellängen richtig, nun aber kämen die Akzente deutlich zu hoch zu liegen, selbst wenn das `german`-Paket benutzt wird. Auch hier waren Änderungen an den METAFONT-Programmen nötig.

Zusätzliche Schriftschnitte

Für eine sinnvolle praktische Verwendung wird natürlich mehr als nur ein einziger Schriftschnitt benötigt. Neben der normalen Variante braucht man wenigstens noch eine kursive oder schräggestellte und eine (halb)fette.

Analog zu CM Sans Serif wurde auf die Erzeugung einer echten Kursiven verzichtet und nur von `cmssi10` der Parameter für die Neigung übernommen; damit war die schräggestellte Variante schon fertig. Für eine fette Schrift, die zur CM Bright paßt, kann man von `cmssbx10` ausgehen und die vertikalen Proportionen aus `cmssdc10` übernehmen, denn alle CM-Schriften weisen das gleiche Formprinzip auf.

Für die Schrift `cmssbx` stehen im Zusammenhang mit den DC-Fonts neuerdings verschiedene Entwurfsgrößen zur Verfügung. Deren Parametersätze wurden aber nicht benutzt, denn die Anpassung von Zeichenbreiten und Strichstärken an die verschiedenen Nenngrößen erschien mir etwas unausgewogen. Der zugehörige *font definition file* von LaTeX2$_\varepsilon$ berücksichtigt diese Entwurfsgrößen übrigens auch nicht.

Das Ergebnis

Abbildung 1 zeigt Beispiele aus der neuen Schriftfamilie „Computer Modern Bright“. Der Name enthält keinen Hinweis auf die fehlenden Serifen, denn der Entwurf einer dazu passenden *Roman*-Variante ist keinesfalls beabsichtigt. Alle Fonts sind nur für eine Nenngröße von 10 pt entworfen, so daß abweichende Schriftgrade beim Rastern durch maßstäbliches Vergrößern oder Verkleinern erzeugt werden müssen. Dies ist eigentlich nicht korrekt, denn bei großen Abweichungen von der Entwurfsgröße sollten Zeichenbreite, Zeichenabstände und Strichstärken deutlich schwächer skaliert werden als die Schrifthöhe. Wie man sieht, fallen die großen Schriftgrade trotzdem noch brauchbar aus. Unterhalb 7 pt kann das Ergebnis aber gerade noch als „nicht unleserlich“ bezeichnet werden; auf eine Wiedergabe wird deshalb verzichtet.

Mathematikschriften

Das Konzept der Computer-Modern-Fonts erlaubt es, mit dem gleichen Satz an Parametern sowohl einen Text- als auch einen dazu passenden Mathematikzeichensatz zu erzeugen. Die Steuerdatei `mathit.mf` erzeugt die Kleinbuchstaben als echte Kursive, also z. B. „a“ statt „a“, wozu die Parameter der CM Bright aber überhaupt nicht passen. Mit einer kleinen Änderung in dieser Datei, `input romanl` statt `input itall`, erreicht man, daß stattdessen eine schräggestellte Mathematikschrift erzeugt wird. Sie verträgt sich mit den vorhandenen mathematischen Symbolzeichensätzen, die somit nicht neu entworfen werden müssen.

DC Bright

Die CM-Bright-Schriften lassen sich auch in der neuen T1- oder DC-Codierung mit 256 Zeichen generieren. Spezieller Anpassungsbedarf besteht bezüglich der neu hinzugekommenen Parameter für die Lage der Akzente; außerdem müssen die Korrekturen bezüglich der i-Punkte und Punkt-Akzente auf die METAFONT-Programme übertragen werden. Versuchsweise wurden diejenigen Änderungen realisiert, welche für die Buchstaben i und j, die damit gebildeten Li-

Abbildung 1: Computer Modern Bright

gaturen und die deutschen Umlaute nötig sind. Das Design der DC-Fonts ist bekanntlich noch nicht stabil, doch sofern keine konzeptionellen Änderungen vorgenommen werden, sollte es möglich sein, auch die Schriftfamilie CM Bright endgültig auf die erweiterte Codierung umzustellen.

Fazit

Trotz zunehmender Verbreitung von PostScript scheint mir auch die Bereicherung des Angebots an METAFONT-basierten Schriften keine „brotlose Kunst“ zu sein. Sicherlich bedarf der hier vorgestellte Schriftentwurf noch umfangreicher Optimierung. Dies betrifft vor allem die Zeichenabstände beim normalen und kursiven Schriftschnitt und den Entwurf zusätzlicher Schriftgrade unterhalb 10 pt. Wer sich darüber ein genaueres Urteil bilden oder vielleicht gar dazu beitragen möchte, aus dem Experiment CM Bright eine praktische verwendbare Schrift zu machen, kann von mir die METAFONT-Quellen erhalten sowie Makropakete für LATEX, die die Nutzung von CM Bright als Grundschrift für komplette Dokumente gestatten.

Literatur

[1] Donald E. Knuth: Computer Modern Typefaces. Addison-Wesley, Reading, Mass. 1986.

[2] Jan Tschichold: Erfreuliche Drucksachen durch gute Typographie. Maro-Verlag, Augsburg 1988 (Nachdruck der Ausgabe von 1960).

Orale Spielereien mit TEX – Teil I

Bernd Raichle

Mit TEX's Makroprozessor, dem „Mund“ oder, wie wir Schwaben sagen würden, der „Gosch“, kann man mehr anstellen, als man auf den ersten Blick vermutet. In dieser neuen Serie soll dieser Makroprozessor etwas genauer unter die Lupe genommen werden. Ich hoffe, daß dies durch einführende Artikel zum Mund, dessen Primitiven und vielen Tips, Tricks und Kniffen in kurzweiliger Form gelingen wird. Obwohl die meisten Artikel für den schon etwas fortgeschritteneren TEX-Anwender geplant sind, sollten die Erläuterungen und Beispiele auch für TEX-Anfänger von Nutzen sein und ihn dazu ermutigen, sich einmal in die (Un-)Tiefen des TEXbook zu vertiefen und zu eigenen Entdeckungsreisen anregen. Wenn man bei diesen Entdeckungsreisen auf Dinge stößt, die auch andere Leser interessieren könnten, sollte man an den armen Autor dieser Serie denken, der sich vorgenommen hat, diese Seiten in jeder Ausgabe zu füllen `:-)`

* * *

Der „Dirty Tricks“-Anhang D des TEXbook [1, S. 373ff] enthält einige Probleme und Problemlösungen, die immer wieder Personen dazu veranlaßt, eine bessere, schnellere, kleinere oder einfach nur elegantere Lösung zu finden und zu veröffentlichen.

Gleich das erste Problem, das in Anhang D beschrieben ist, scheint sehr interessant zu sein, da im Laufe der Zeit mehrere verbesserte Lösungen gefunden wurden:

> Gegeben sei eine nichtnegative, ganze Zahl in einem Register `\n`. Definiere ein Makro `\asts`, das zu `\n` Sternchen `*` expandiert.

Will man das Makro `\asts` innerhalb einer `\message`- oder `\write`-Anweisung verwendet, liegt das Problem dieser Aufgabe darin, daß man in der Definition des Makros `\asts` bis auf die Sternchen nur expandierbare TEX-Primitive verwenden darf. Die einfache Lösung

```
\def\asts{{\loop \ifnum\n>0 *\advance\n -1 \repeat}}
```

führt bei `\message{\asts}` zu Fehlermeldungen.

Natürlich findet man im TEXbook gleich einige Lösungen dieses Problems.

```
\begingroup
  \xdef\asts{}
  \loop
    \ifnum\n>0
      \xdef\asts{\asts *}
      \advance\n -1
  \repeat
\endgroup
```

In dieser einfachen Lösung wird `\asts` zu Beginn initialisiert und anschließend hängt jeder Schleifendurchlauf ein zusätzliches Sternchen an die Definition des Makros `\asts`. Leider hat diese Lösung eine Komplexität $O(n^2)$, so daß bei der Generierung der doppelten Sternchenzahl die vierfache Zeit benötigt wird.

Wie erwartet, findet man im TEXbook auch eine schnellere Lösung mit Komplexität $O(n)$.

```
\begingroup
  \aftergroup\def\aftergroup\asts\aftergroup{
    \loop \ifnum\n>0 \aftergroup* \advance\n -1 \repeat
  \aftergroup}
\endgroup
```

Das ganze Know-How dieser Lösung steckt im Primitiv `\aftergroup`. Mit diesem Primitiv wird die Makrodefinition der Definition

```
\def\asts{**...***}
```

Token für Token auf einen Stack gelegt, der am Ende der Gruppe abgearbeitet wird und damit das Makro `\asts` mit der gewünschten Sternchenzahl definiert. Der Nachteil dieser Lösung: für jedes `\aftergroup` wird ein Eintrag im Input-Level-Stack und ein Eintrag im Savestack benötigt. Da der Input-Level-Stack bei den meisten Implementierungen auf 200 Einträge begrenzt ist, tritt bei mehr als 180–190 Sternchen ein Überlauf dieses Stack auf.

Soweit zwei der drei im TEXbook vorgestellten Lösungen. Im TUGboat-Artikel [2] stellt George Russell zwei Lösungen vor, die weitere Ideen zur geschickteren Realisierung vorstellen. Die erste Lösung, die wie die Knuth'schen Lösungen mit Zuweisungen arbeitet, nutzt die Idee, statt an die Definition des Makro nur ein Sternchen pro Schleifendurchlauf hinzuzufügen, möglichst die Sternchenzahl in jedem Schritt zu verdoppeln. So werden weniger Schritte benötigt und aus dem quadratischen Laufzeitverhalten wird ein lineares, ohne die Nachteile des Stack-Überlaufs, die der `\aftergroup`-Lösung innewohnt.

Für diese Artikelserie stellt die zweite Lösung die interessantere dar. Sie ist endlich voll expandierbar und benutzt dabei sehr geschickt die TEX-Primitive `\csname` und `\the` mit einigen wenigen, aber sehr effektiv definierten Makros. Hier ist die Lösung aus dem genannten TUGboat-Artikel, die ich zum leichteren Verständnis an wenigen Stellen abgeändert habe:

```
\def\endofnumber#1{#2}
\def\0{\nextdigit{}}       \def\5{\nextdigit{*****}}
\def\1{\nextdigit{*}}      \def\6{\nextdigit{******}}
\def\2{\nextdigit{**}}     \def\7{\nextdigit{*******}}
\def\3{\nextdigit{***}}    \def\8{\nextdigit{********}}
\def\4{\nextdigit{****}}   \def\9{\nextdigit{*********}}

% #1 Sternchen fuer aktuelle Ziffer
% #2 Sternchen - Zwischenergebnis (muessen verzehnfacht werden)
% #3 naechste Ziffer (oder 'endofnumber')
\def\nextdigit #1#2#3{%
  \csname #3\endcsname {#1#2#2#2#2#2#2#2#2#2#2}}

\def\Asts#1{\nextdigit{}{}#1{endofnumber}}
\edef\asts{\expandafter\Asts\expandafter{\the\n}}
```

Die Idee hinter diesen Makros ist einfach: Die Zahl im Register `\n` wird durch das Primitiv `\the` in `\asts` in die einzelnen Ziffer-Tokens umgewandelt. Aus

der Zahl 13 werden bei diesem Schritt die Ziffer-Tokens `1` und `3` erzeugt. Das Makro `\nextdigit` liest diese Ziffern nacheinander und erzeugt dabei mit Hilfe des Primitivs `\csname` und den `\0`... `\9`-Makros die entsprechende Anzahl an Sternchen. Da jede Ziffer einer Zehnerstelle entspricht, wird dabei die bisherige Sternchenzahl verzehnfacht.

Am anschaulichsten wird dies an einem Beispiel. Für `\n` = 13 ergibt sich folgende Expansionsfolge:

```
\asts
\expandafter\Asts\expandafter{\the\n}
\Asts{13}
```

Das Makro `\nextdigit` hat drei Parameter: Die ersten beiden Parameter enthalten die Sternchen für die aktuelle Ziffer und für das bis dahin „errechnete“ Zwischenergebnis. Zu Beginn enthalten beide keine Sternchen. Im nächsten Expansionsschritt liest `\nextdigit` die Ziffer `1` als drittes Argument.

```
\nextdigit{}{}13{endofnumber}
\csname 1\endcsname {}3{endofnumber}
\1{}3{endofnumber}
```

Die jeder Ziffer zugeordneten Makros `\0`... `\9` führen zu einer erneuten Expansion des Makros `\nextdigit`, so daß mit der dadurch gebildeten Schleife die Ziffern nacheinander gelesen werden.

```
\nextdigit{*}{}3{endofnumber}
\csname 3\endcsname{*}{endofnumber}
\3{*}{endofnumber}
```

Der nun folgende Expansionsschritt macht die Idee hinter der ziffernweise Abarbeitung deutlich: Die ersten beiden Parameter entsprechen der Sternchenzahl der Zehner- und der Einerstelle. Um nun die den beiden Stellen entsprechende Sternchenzahl zu erhalten, müssen einfach nur die Sternchen der Zehnerstelle in `#2` verzehnfacht und die Sternchen der Einerstelle in `#1` angehängt werden.

```
\nextdigit{***}{*}{endofnumber}
\csname endofnumber\endcsname {**********}
\endofnumber{**********}
**********
```

In der gezeigten Form kann man mit dem Makro `\asts` mit einem normal großen TEX bis zu ca. 27 000 Sternchen erzeugen. Die doppelte Zahl ist mit etwas anderen Makrodefinitionen möglich, wie sie z. B. im Originalartikel verwendet wurden.

Ein etwas anderer Ansatz für eine voll expandierbare Lösung des `\n`-Sternchenproblems verwendet Sonja Maus [3]. Ihre Lösung zeigt, wie man Zahlen, die durch `\the` oder `\number` in Ziffertokens umgewandelt vorliegen, durch einfache Textersetzung in TEX's Mund halbieren kann. Die Idee, die in der dadurch voll expandierbaren Lösung von Sonja Maus und in der mit Zuweisungen arbeitenden ersten Lösung von George Russell steckt, ist bis auf diesen Unterschied dieselbe. Ich werde nochmals in einer zukünftigen Folge dieser Serie genauer auf die Lösung von Maus eingehen.

Nach diesen Artikeln aus dem Jahre 1991 gab es lange Zeit keine weiteren Lösungen, die neue Ideen verwendet haben. Im Dezember 1993 veröffentlichte David Kastrup über die Newsgroup `comp.text.tex` eine neue Lösung – in Form eines Zweizeilers!

Die voll expandierbare Lösung [4] in zwei Zeilen

```
\def\convertmtoast#1{\if m#1*\expandafter\convertmtoast\fi}
\edef\asts{\expandafter\convertmtoast\romannumeral\number\n 001}
```

zeigt, daß man oft mit Primitiven, die auf den ersten Anschein nichts zur Lösung einer Aufgabe beitragen können, sehr elegant, kompakt und schnell Probleme lösen kann.

Der dabei verwendete Trick ist einfach: Das Primitiv `\romannumeral` erzeugt aus einer fast beliebig großen Zahl die entsprechende römische Ziffernfolge in Kleinbuchstaben. Da 1000 die größte römische Zahl mit einer Ziffer, hier `m`, darstellt, werden für Zahlen größer 1000 entsprechend viele `m` erzeugt. Um dies ausnutzen zu können, muß die Zahl in `\n` nur mit 1000 multipliziert und anschließend die entstehenden `m` durch Sternchen ersetzt werden.

Ein Beispiel mit der Expansionsfolge für `\n` = 13 soll dies wieder anschaulich machen:

```
1 \asts
2 \expandafter\convertmtoast\romannumeral\number\n 001
```

Als Argument für das Primitiv `\romannumeral` werden alle nachfolgenden Ziffern als umzuwandelnde Zahl gelesen. Dabei werden alle Tokens expandiert. In unserem Fall expandiert `\number\n` zu den Ziffern `13`, die von `\romannumeral` mit den nachfolgenden drei Ziffern `001` als *eine* Zahl gelesen werden.

```
3 \expandafter\convertmtoast\romannumeral 13001
```

Nach der Umwandlung werden die Buchstaben `m` durch das Makro `\convertmtoast` durch Sternchen ersetzt. Die Expansion des Makros

`\convertmtoast` erfolgt in mehreren Schritten, die hier in einem Expansionsschritt zusammengefaßt dargestellt werden.

```
4 \convertmtoast mmmmmmmmmmmmmi
5 *\convertmtoast mmmmmmmmmmmmmi
6 **\convertmtoast mmmmmmmmmmmmi
```

Dieser Schritt wird für alle `m` durchgeführt.

```
17 *************\convertmtoast i
```

Der einzelne Buchstabe `i` am Ende beendet die Ersetzung durch Sternchen. Um diesen Buchstaben zu erzeugen, wurde deshalb die Zahl, die `\romannumeral` liest, neben der Multiplikation mit 1000 noch um 1 erhöht.

```
18 *************
```

Diese Lösung ist etwas langsamer als die zuvor vorgestellte, ebenso voll expandierbare Lösung von George Russell. Sie wird jedoch nicht durch TEX's „main memory", sondern durch den freien Platz in TEX's „string pool" begrenzt, so daß man normalerweise ohne Probleme 5000–20 000 Sternchen erzeugen kann.

Beschränkt man sich nun nicht mehr darauf, daß die Lösung, mit der man `\asts` definiert, voll expandierbar sein muß, so kann man zur Umwandlung der `m` in Sternchen eine etwas schnellere Lösung benutzen. Diese Definition stammt von Éamonn McManus aus den „followups" [5] auf David Kastrups Beitrag.

```
\begingroup \uccode`\m=`\*
\uppercase\expandafter{\expandafter\endgroup
\expandafter\def\expandafter\asts
  \expandafter{\romannumeral\the\n 000 }}
```

Um die vielen `\expandafter` zu vermeiden, konnte ich mir eine weitere, kleine Änderung an diesen Definitionen nicht verkneifen.

```
\begingroup \uccode`\m=`\*
\def\x{\uppercase\bgroup\endgroup \def\asts}
\expandafter\x\expandafter{\romannumeral\the\n 000 }}
```

Versucht sich jemand an einer Erklärung, welche Klammerpaare bei der Abarbeitung dieser Eingabe zusammengehören und in welchen Schritten das Makro `\asts` definiert wird?

* * *

Um die erste Folge der neuen Serie zu beschließen, möchte ich noch eine LaTeX-Änderung vorstellen, die durch Davids `\romannumeral`-Lösung inspiriert wurde und die in manchen Fällen nützlich sein kann.

In LaTeX kann man durch `\Roman{foo}` den Zähler `foo` in großen römischen Ziffern darstellen. Leider erzeugt nun

```
\newcounter{foo}
\setcounter{foo}{14}
\typeout{Wert=\Roman{foo}}
```

nicht die Ausgabe `Wert=XIV`, sondern `Wert=\uppercase {xiv}`, während die Verwendung von `\roman` das erwartete Ergebnis ausgeben würde. In der Definition des Makros `\Roman` wird das nichtexpandierbare Primitiv `\uppercase` verwendet, das erst beim Ausführen die Tokens im Argument in Großbuchstaben umwandelt. Eigentlich ist es sogar ein Glück, daß `\Roman` nicht *zerbrechlich* ist!

Mit den zwei folgenden Makros erhält man ein voll expandierbares `\Roman`.

```
\newcommand{\convertrtoR}[1]{%
  \if i#1I\else\if x#1X\else\if c#1C\else\if m#1M\else
  \if v#1V\else\if l#1L\else\if d#1D\fi\fi\fi\fi\fi\fi\fi
  \ifx\relax #1\else \expandafter\convertrtoR\fi}
\renewcommand{\@Roman}[1]{%
  \expandafter\convertrtoR\romannumeral #1\relax}
```

Da man das interne Makro `\@Roman` umdefiniert, müssen diese Makros in eine `.sty`-Datei geschrieben oder in der Dokumentpräambel mit `\makeatletter` ... `\makeatother` geklammert definiert werden.

In der nächsten Folge geht es dann mit einem kleinen TEX-Anatomiekurs weiter.

Literatur

[1] Donald E. Knuth: *The TEXbook*, Addison-Wesley Publ., Reading, Mass., Juni 1991.

[2] George Russell: Generating `\n` asterisks, *TUGboat*, 12(1991), No. 2, S. 278 f.

[3] Sonja Maus: An Expansion Power Lemma, *TUGboat*, 12(1991), No. 2, S. 277.

[4] David Kastrup: TEXbook's Appendix D outdirted—considerably, Beitrag in der *Newsgroup* `comp.text.tex`, 8. Dezember 1993.

[5] Éamonn McManus: „Followup“ auf [4] in der *Newsgroup* `comp.text.tex`, 9. Dezember 1993.

Paralleles Setzen längerer Texte

Matthias Eckermann

Im Leben eines Latein- und Theologiestudenten ergibt sich irgendwann einmal folgendes Problem: ein längerer Text ist in zwei Sprachen möglichst parallel zu setzen. Dabei sollen auch die Seitenumbrüche automatisch erfolgen, der Schreibende nicht gezwungen sein, diese Arbeit mit Hilfe von `minipage`s oder ähnlichen Konstruktionen „nachzubauen“; auf dieses Problem wird auch im „Offizin“ [1, S. 22–25] eingegangen. Günter Koch stellt dort drei Anforderungen:

- Statt *einer* Menge durchgängig gezählter Fußnoten soll es auch deren zwei geben können – pro Text eine.
- Die Fußnoten sollen unter der Spalte stehen, zu der sie gehören, oder, wenn sie die Breite der ganzen Seite einnehmen sollen, in zwei getrennten Apparaten gesammelt gesetzt werden.
- Statt in gegenüberliegenden Spalten sollte man Texte auch auf gegenüberliegenden Seiten parallel setzen können.

Ich will und kann nicht behaupten, die Probleme letztgültig und befriedigend gelöst zu haben, doch glaube ich, meine Lösung vorstellen zu sollen, damit sich vielleicht ein genialer TEXniker findet, der *mir* Unmögliches möglich macht. Immerhin ist von Günter Kochs Wünschen der erste leicht erfüllbar, der zweite (mit meinem Ansatz) nicht, der dritte bedarf noch einiger Mühe.

Benutzung

Nach einem ausgiebigen Kampf mit TEXs `box`-Anweisungen [2, 3] entstand im Oktober, kurz vor der Tagung in Katlenburg-Lindau, eine erste vollständig unbefriedigende Version. Dank verschiedener Ratschläge war ich in der Lage, bis jetzt noch einiges zu verbessern.

In der Datei `parallel.sty` findet sich der folgende Hinweis zum Gebrauch:

```
%   \begin{Parallel}{<left-width>}{<right-width>}
%     \ParallelLText{<left-text>}
%     \ParallelRText{<right-text>}
%   \end{Parallel}
```

Die Umgebung `Parallel` wird mit einer festen linken und rechten Breite aufgerufen: das ist durchaus sinnvoll, da zwei Sprachen – wenn es sich um eine Übersetzung handelt – stets in etwa das gleiche „Volumen"-Verhältnis zueinander haben[1].

Innerhalb der Umgebung werden die jeweiligen Texte für links und rechts als Argumente zweier Makros übergeben. Leider ist die Länge von Makroparametern durch TEX begrenzt; es kann daher bei sehr langen Texten zu Problemen mit `<left-text>` bzw. `<right-text>` kommen[2]. Doch kann diese Schwierigkeit durch die explizite Eingabe von Absätzen (und welcher Text hätte solche nicht?) umgangen werden:

```
%   \begin{Parallel}{<left-width>}{<right-width>}
%     \ParallelLText{<left-text>}
%     \ParallelRText{<right-text>}
%     \ParallelPar
%     \ParallelLText{<left-text>}
%     ...
%   \end{Parallel}
%             Thanks to H.Kopka, B.Raichle, J.Schrod.
```

Nach dem Befehl `\ParallelPar` für *paralleler Paragraph* beginnt die Ausgabe der Spalten wieder auf gleicher Höhe: die Gesamtlänge eines Absatzes richtet sich folglich nach dem längeren Text. Innerhalb der `Parallel`-Umgebung können beliebig viele solcher Absätze stehen.

Funktionsweise des neuen Fußnoten-Makros

Wenn der Text als `<left-text>` bzw. `<right-text>` eingelesen wird, wird er sofort – gemäß der gesetzten Textbreite für links bzw. rechts – in entsprechende `\vbox`es „aufgeteilt". Dabei gehen die Fußnoten entweder verloren oder sie erscheinen (mit `\protect\footnote`) direkt im Text: beides ist inakzeptabel. Um das zu umgehen, habe ich den Befehl `\footnote` innerhalb der `Parallel`-Umgebung durch eine neue Definition mit folgender Funktion ersetzt:

[1] „Volumen" bezogen auf die Zeichenmenge. Latein – Deutsch (Prosa) haben z. B. ein Verhältnis von etwa 4:5.

[2] Mit emTEX (`tex386`, main memory = 500 000) und vermutlich auch anderen großen Versionen von TEX ist das bei einer Größe von mehr als 50 000 Zeichen kein Problem.

– Der Fußnotentext wird Inhalt eines Makros.

– Dieses Makro erhält den Namen `\ParallelFootNote`*xx*, wobei *xx* für den aktuellen Stand des Fußnotenzählers in römischen Ziffern steht.

– Beim Ausdrucken der Fußnoten am Ende der Umgebung werden diese Makronamen in gleicher Weise wieder produziert, die Makros werden aufgerufen, die Fußnotentexte ausgegeben.

– Bei dieser Verwirklichung der Fußnoten geht allerdings die Information verloren, auf welcher *Seite* eine Fußnote steht; es läßt sich nur noch bestimmen, in welchem *Absatz* sich eine bestimmte Fußnote befindet.

Das Makro `\ParallelFootnote` sieht dann so aus:

```
\newcommand{\ParallelFootnote}[1]%
  {\global\advance\ParallelFNCounter by 1%
   \footnotemark[\number\ParallelFNCounter]%
   \expandafter\gdef
     \csname ParallelFootNote%
       \romannumeral\ParallelFNCounter\endcsname{#1}}
```

Beispiel: Augustinus, De Trinitate V, viii (9) – V, ix (10)

[VIII 9] Quapropter[1] illud praecipue teneamus, quidquid ad se dicitur praestantissima illa et divina sublimitas substantialiter dici; quod autem *ad aliquid*[2] non substantialiter sed relative; tantamque vim esse *eiusdem substantiae*[3] in patre et filio et spiritu sancto ut quidquid de singulis ad se ipsos dicitur non pluraliter in summa sed singulariter accipiatur. Quemadmodum enim *deus* est *pater* et *filius deus* est *et spiritus sanctus deus* est, quod secundum substantiam dici nemo dubitat, *non tamen tres deos sed unum deum*[4]

[VIII 9] Wollen wir also vor allem daran festhalten, daß alle Aussagen, die von der herrlichen, göttlichen Erhabenheit in bezug auf sie selbst gemacht werden, die Substanz betreffen, alle Aussagen hingegen, welche in bezug auf etwas gemacht werden, nicht die Substanz, sondern die Beziehung betreffen, und daß die Kraft der gleichen Substanz im Vater, Sohne und Heiligen Geiste so groß ist, daß alles, was von den einzelnen Personen in bezug auf sie selbst ausgesagt wird, von ihnen zusammen nicht in der Mehrzahl, sondern nur in der Einzahl ausgesagt werden darf. Wie nämlich der Vater Gott ist, der Sohn Gott ist, der Heilige Geist Gott ist, so – niemand zweifelt daran, daß es sich hier um Aussagen hinsichtlich der Substanz

dicimus eam ipsam praestantissimam trinitatem. Ita magnus pater, magnus filius, magnus et spiritus sanctus; nec tamen tres magni sed unus magnus. Non enim de patre solo sicut illi perverse sentiunt, sed de patre et filio et spiritu sancto scriptum est: *Tu es solus deus, magnus*[5] Et bonus pater, bonus filius, bonus et spiritus sanctus; nec tres boni sed unus bonus de quo dictum est: *Nemo bonus nisi unus deus.*[6]

handelt – heißen wir diese erhabene Dreieinigkeit doch nicht drei Götter, sondern nur einen[7] Gott. Ebenso ist der Vater groß, der Sohn groß und der Heilige Geist groß, doch sind es nicht drei große, sondern es ist nur ein großer. Denn nicht vom Vater alleine, wie die Häretiker in ihrer verkehrten Denkweise glauben, sondern vom Vater, Sohne und Heiligen Geiste gilt das Schriftwort: „Du allein bist groß, o Gott.“ Ferner ist der Vater gut, ist der Sohn gut, ist der Heilige Geist gut, und doch sind nicht drei Gute, sondern ein Guter, von dem es heißt: „Niemand ist gut als Gott allein.“

Probleme und Desiderata

Wer diese Umgebung und Makros benutzt, wird schnell merken, woran es fehlt:

- Innerhalb von `<left-text>` bzw. `<right-text>` erzeugen Umgebungen wie `tabular` und `minipage` „unexpected results“. Außerhalb der Parameter, jedoch innerhalb der Umgebung sollten sie zu den gewünschten Ergebnissen führen.
- Fußnoten werden leider erst am Ende der Umgebung `Parallel` ausgegeben. Es ist zwar möglich, die Fußnotenausgabe mit dem Absatzumbruch `\ParallelPar` zu kombinieren. Ich weiß aber nicht, ob das sinnvoller ist.
- Der Zeilenumbruch erfolgt vollständig parallel, d. h. Größenunterschiede der Zeilen links und rechts schlagen sich in ungleichmäßigen Zeilenabständen nieder. Ich halte das aber für vertretbar, weil diese auffallenden

[1] Wenn man nicht über lateinische Trennmuster verfügt (CTAN: tex-archive/language/hyphenation/lahyph.tex), sollte man besser ganz auf Trennung verzichten ...

[2] ... cf. Arist., categ. VII. 6a, 36-8b, 24; Ps.-Aug., categ. X ex Arist., XI (PL 32, 1430)

[3] Symb. Nicaen., 10

[4] vide Symb. ‘Quicumque,’ 17-18 [i.e. DS 75-76]; cf. Ambr., spir III, xvi, 109 (CSEL 79, 196, 3/5)

[5] Ps. 85, *10

[6] Marc 10,18; Luc 18, *19

[7] Ein provozierter „Fehler“, der mögliche ungleichmäßige Zeilenabstände veranschaulichen soll.

Differenzen bei den klassischen Anwendungen parallelen Setzens nicht vorzukommen scheinen.

– Ein paralleles Setzen auf zwei gegenüberliegenden Seiten ist zwar (mit Modifizierungen) möglich, aber (noch?) nicht im mindesten elegant und TEX angemessen.

Literatur

[1] DANTE e.V. (Hg.), Peter Scherber (Bearb.): Offizin, Addison-Wesley, Bonn, 1994.

[2] Schwarz, Norbert: Einführung in TEX, 2. überarbeitete Auflage, Addison-Wesley, Bonn, 1988.

[3] Kopka, Helmut: LATEX – Erweiterungsmöglichkeiten, Addison-Wesley, Bonn, 1990.

Was Sie schon immer über TEX wissen wollten

. . .

Absatzformen

Luzia Dietsche

Auch diesmal stammt die Idee für den Tip aus einer Anfrage über Diskussionslisten. Allerdings dreht es sich hierbei nicht wie sonst um ein Problem, von dem ich vermute, daß sich schon oft jemand damit herumgeschlagen hat, sondern eher um ein etwas exotischeres Begehren. An der (einfachen) Lösung kann man aber meiner Meinung nach gut sehen, daß (fast?) jede Aufgabenstellung mit LATEX bzw. TEX bewältigt werden kann.

Die ursprüngliche Frage lautete sinngemäß, wie man die letzte Zeile einer Abbildungsunterschrift zentrieren könne.

Die „grobe“ Antwort darauf war (basierend auf einer Lösung von Anne Brüggemann-Klein [1]):

> The most elegant way avoids fiddling with boxes, but is counterintuitive (elegance = counterintuitiveness very often in TEX!!)
>
> ```
> \leftskip=0pt plus 1fil
> \rightskip=0pt plus -1fil
> \parfillskip=0pt plus 2fil
> ```
>
> Take the definition of `\@makecaption` from the appropriate style file, put `\begingroup` and the two lines above at the very beginning; put `\endgroup` at the end.

Eben das habe ich getan, wobei ich auf die Klammerung der Definition durch `\begingroup` und `\endgroup` verzichtet habe. Die Definition von `\@makecaption` findet sich in `article/report/book.cls`, wobei es egal ist, welche der drei Dateien man verwendet. Bei allen dreien ist die Definition die gleiche. Nun darf man natürlich nicht im Original eingreifen und ändern. Deshalb kopiert man alles, was zwischen `\long\def\@makecaption` und `\belowcaptionskip\fi}` steht, in eine eigene Datei, die die Endung `.sty` hat. In diese Definition fügt man nun den Dreizeiler von oben ein und schon hat man das gewünschte Ergebnis. Verfügbar macht man sich die Änderung, indem man die Datei über die Anweisung `\usepackage{`⟨*Datei*⟩`}` einbindet.[1]

[1] Weitere Variationen zur `\caption`-Anweisung findet man in [2, S. 157f] beschrieben.

Das ist die Vorgehensweise für LaTeX 2ε. Wenn man aber eine Definition zur Verfügung stellen will, die sowohl mit LaTeX 2ε als auch mit LaTeX 2.09 funktioniert, muß man noch eine zusätzliche Abfrage einbauen. Anstelle der Zeile

```
\vskip\abovecaptionskip
```

fügt man die folgende Abfrage ein:

```
\@ifundefined{abovecaptionskip}%
  {\vskip 10pt}%
  {\vskip\abovecaptionskip}%
```

Damit überprüft LaTeX, ob das Längenmaß `\abovecaptionskip` definiert ist oder nicht. Falls es nicht definiert ist, geschieht nichts, ansonsten wird ein vertikaler Abstand mit diesem Maß eingesetzt. Die gleiche Abfrage wird nochmal am Ende der Definition mit `\belowcaptionskip` vorgenommen.

Nimmt man all dies zusammen, sieht die (Beispiels-)Datei `newcap.sty` wie folgt aus:

```
\long\def\@makecaption#1#2{%
  \@ifundefined{abovecaptionskip}%
    {\vskip 10pt}%
    {\vskip\abovecaptionskip}%
  \leftskip=0pt plus 1fil        % neu
  \rightskip=0pt plus -1fil      % neu
  \parfillskip=0pt plus 2fil     % neu
  \sbox\@tempboxa{#1: #2}%
  \ifdim \wd\@tempboxa >\hsize
    #1: #2\par
  \else
    \global \@minipagefalse
    \hbox to\hsize{\hfil\box\@tempboxa\hfil}%
  \fi
  \@ifundefined{belowcaptionskip}%
    {}%
    {\vskip\belowcaptionskip}}

\endinput
```

Die Anweisung `\endinput` sollte am Ende jeder eingebundenen Datei stehen. Mit ihr ist für LaTeX das Ende der Datei erreicht, gleichgültig was noch folgt.

Nun will man vielleicht den Effekt mit der zentrierten letzten Zeile auch in einem Absatz verwenden. Das ist natürlich kein Problem. So kann man sich zum Beispiel eine Umgebung definieren, bei der nur zu Beginn und Ende ein Absatz gemacht und die letzte Zeile eines Absatzes automatisch zentriert wird:

```
\newenvironment{ZenZeil}{%
  \par
  \leftskip=0pt plus 1fil
  \rightskip=0pt plus -1fil
  \parfillskip=0pt plus 2fil
  \relax                      % wichtig!
}{%
  \par
}
```

Und wenn man diese Umgebung verwendet, wird der darin enthaltene Absatz so formatiert wie der, den Sie gerade eben lesen. Die hier vorgestellten Makros sind nicht vollständig durchgetestet und sollen wie immer einen Ansatz bieten, um solche oder ähnliche Problemstellungen lösen zu können.

Literatur

[1] Anne Brüggemann-Klein: Obtaining a funny paragraph shape in TEX, TEXhax Vol. 89, Issue 42, Mai 1989.

(CTAN: `tex-archive/digests/texhax/89/`)

[2] M. Goosens, F. Mittelbach, A. Samarin: Der LATEX-Begleiter, Addison-Wesley, Reading, Mass., 1994.

TEX-Beiprogramm

WinWord versus LaTeX

Horst Peiffer

Anmerkung der Redaktion: In den diversen Kommunikationsmedien kommt es in regelmäßigen Abständen zu mehr oder weniger heftig geführten Diskussionen über Pro und Contra von TEX und sogenannten WYSIWYG-Systemen. Da es scheinbar sehr viele Personen gibt, die freiwillig oder gezwungen mit beiden Arten der Textverarbeitung arbeiten, bat ich während einer dieser Diskussionen um Erfahrungsberichte. Dieser Artikel ist das Ergebnis der Anfrage. Ich würde mich freuen, wenn es sich hierbei nur um einen Anfang handeln würde ...

Luzia Dietsche

Über allen (Vor-)Urteilen

Die Entscheidung für oder gegen eine Software ist doch grundsätzlich mit der Frage verknüpft, wo man sie hernimmt und was sie kostet. Die einfache und schnelle Antwort lautet in der Regel, irgendwo kopieren, wobei sich das Kostenproblem gleichzeitig erledigt. Damit hat man aber, und das muß hier unbedingt betont werden, im Falle einer kommerziellen Software (sprich WinWord) bereits einen Fuß im Knast. Alle, die jetzt hier aufstöhnen und mit süffisantem Unterton mitteilen, daß sie niemals geklaute Software, sondern höchstens übervollständige Demoversionen benutzen, seien gewarnt: Verstöße gegen das Urheberschutzgesetz sind Straftaten. Wenn man erwischt wird, führt das fast immer zu einer Vorstrafe. Was das wiederum zum Beispiel für eine Promotion bedeutet, läßt sich in den einschlägigigen Ordnungen nachlesen: in aller Regel das Ende aller „Dr.“-Träume! Und das ist nur ein Punkt. Auch viele Arbeitgeber sind gegenüber Vorstrafen ihrer Mitarbeiter äußerst empfindlich, nicht nur bei der Stellensuche oder beim -wechsel, sondern auch bei bestehenden Arbeitsverhältnissen.

Ein Risiko berechnet sich grundsätzlich aus Schadenswahrscheinlichkeit (hier: das Risiko, erwischt zu werden) und Schadensumfang (hier: persönliche Konsequenzen). Mag der erste Punkt hier auch klein sein, der zweite ist unter Umständen gewaltig, d. h. das Risiko ist beträchtlich.

Unter Berücksichtigung der Kosten für den legalen Erwerb ist für eine Privatperson die Entscheidung zwischen WinWord (i. a. kommerzieller Textverarbeitung) und LaTeX normalerweise bereits hier gefallen.

Die Wahrheit über LaTeX

Da wir hier unter uns sind, brauche ich mich wohl nicht lange über die Vorteile von LaTeX auszulassen. Wir wissen, warum wir dieses System bevorzugen, aber obwohl ich selber TeXoholic bin, kann ich den Fanatismus und den missionarischen Eifer einiger TeXnics nicht nachvollziehen. Sicher, TeX/LaTeX ist *das* Mittel der Wahl bei komplexen Dokumenten insbesondere technisch-wissenschaftlicher Richtung[1], aber wo viel Licht ist, ist auch viel Schatten:

- Die Benutzerführung ist auf den ersten Blick abschreckend. Da es für den ersten Eindruck nun mal keine zweite Chance gibt, sind die verschiedensten Ressentiments im Grunde verständlich. Daß bei WYSIWYG-Textverarbeitungen viel Blenderei im Spiel ist, sei gar nicht verschwiegen. Trotzdem – bei den meisten Leuten ist der erste Eindruck der entscheidende – 1:0 für WinWord!
- Man mag darüber jammern, aber es ändert nichts: Befehlseingabe über die Tastatur ist megaout – Klicken ist in! Sinnvoll oder nicht (eigentlich habe ich die Hände ja auf der Tastatur, während ich nach der Maus immer erst greifen muß . . .) – 2:0 für WinWord.
- LaTeX verlangt vom Benutzer Vertrauen in den Compiler, WinWord gibt zumindest einen brauchbaren Soforteindruck vom erstellten Dokument, insbesondere bei eingebundenen Tabellen, Zeichnungen usw. Obwohl LaTeX mein Vertrauen kaum jemals enttäuscht hat – 3:0 für WinWord.
- Wenn man nicht von irgendwoher eine lauffähige TeX-Version per Backup ziehen kann, ist man als Laie ganz schön angeschmiert. Obwohl keinesfalls technischer Nichtschwimmer habe ich seinerzeit die Neuinstallation von den Originaldisketten aufgegeben und mir ein Backup einer fertigen Installation besorgt. Wie man den zum Teil verzweifelten Anfragen sogar von Systemspezialisten in den einschlägigen Mailboxen entnehmen kann, ist die Installation tatsächlich hartes Brot – 4:0 für WinWord.

Demgegenüber stehen natürlich auch zahlreiche Vorteile von LaTeX, aber wie gesagt: hier bin ich der *Advocatus Diaboli*, d. h. *Advocatus Microsoftis*.

[1] Ich selbst bin Physiker, also ein prädestinierter TeXer.

Ich hör' auch schon die Kommentare, die sich meist zusammenfassen lassen unter:

Versuch das doch mal mit WinWord! ...

Dazu muß 'mal folgendes grundsätzlich gesagt werden:[2] WinWord hat inzwischen einen Stand erreicht, von dem man mit Fug und Recht behaupten kann, daß er mit all den heißgeliebten und bis vor nicht allzu langer Zeit exklusiven TeX-Features mithalten kann (automatisches Referenzieren, Listen usw. usw.). Allerdings ist das dann auch alles andere als einfach oder gar selbsterklärend. Und das ist der eigentliche Punkt: komplexe Dokumente mit WinWord erstellen und pflegen ist nach wie vor ein Kreuz. Beim ersten Kontakt wird eine Benutzerfreundlichkeit vorgetäuscht, die sich im weiteren als Illusion entpuppt. Man vergleiche nur die 1000 Seiten WinWord-Handbuch mit den 300 Seiten des Kopka.

Das Microsoft-Konzept ist aber trotzdem so erfolgreich, weil es nach dem Prinzip des ersten Eindrucks vorgeht. Wenn man die Leute dazu gebracht hat, Textverarbeitung mit WinWord gleichzusetzen, hat man gewonnen, egal welcher Mist sich hinter der bunten Oberfläche verbirgt.

Insofern muß der Leidensdruck schon außergewöhnlich groß sein, um jemanden von WinWord zu TeX zu treiben. Dies ist aber eben nur bei wirklich komplexen Dokumenten der Fall. Wenn im wesentlichen Geschäftskorrespondenz durch den Rechner geht, wird das überlegene Outfit der WYSIWYGs immer den Vorzug erhalten. Hier führen die Dinger auch Laien in der Regel schnell zum Erfolg oder zumindest zum Ausdruck ...

Außerdem muß beachtet werden, daß es für die WYSIWYG-Anbieter auch um Milliardensummen geht. Hier wird freiwillig keine Handbreit Boden abgegeben. Daß sich auch unter Profis das Konzept einer Markup-Word orientierten Textverarbeitung großer Akzeptanz erfreut, kann man jetzt gerade wieder im WWW beobachten: HTML heißt hier der *State-of-the-Art*, was im wesentlichen nichts anders ist als eine kommerzialisierte (und erweiterte) Fassung des TeX-Konzeptes.

[2] Meine Erfahrungen mit WinWord beziehen sich dabei auf die Version 2.0b, mit der ich bis Ende 1993 an einem Universitätsinstitut gearbeitet habe. Heute arbeite ich unter UNIX mit FrameMaker oder, wenn's eben geht, wie schon bei meiner Diplomarbeit, mit TeX.

Was denn nun?

Zusammenfassend muß gesagt werden, daß es hierzu wohl soviel Meinungen wie Beteiligte gibt. Ich kann dazu nur sagen: Macht doch was Ihr wollt! Ihr werdet schon sehen, was Ihr davon habt, im Guten wie im Bösen ...

Bleibt für jeden persönlich nur die Frage offen, ob jemand, der seine Textverarbeitung verschenkt, überhaupt böse sein kann.

TEXIT! for OS/2

Michael Schank

Die erste Beta-Version für die Bedienung von emTEX im 32-Bit OS/2 Betriebssystem ist von mir selbst herausgegeben worden. Das Programm *TEXIT!*, Version 0.36 ist zum täglichen Arbeiten mit TEX und LATEX auf der OS/2 Workplace-Shell als Presentation-Manager-Programm verfügbar.

Das Programm wird über einen Notebook voll konfiguriert und kann folgende Dinge per Maus-Klick tun:

- Arbeitsdatei auswählen
- Editieren mit externem Editor
- TEX- und LATEX-Übersetzungen in Deutsch und Englisch
- PreView mit Full-Screen und PM-View Programm
- Drucken mit diversen einstellbaren Werten (z. B. Druckbereich, ...)
- Konfiguration von *TEXIT!* und Aufruf eines Programms namens *FileCommander* (Norton Commander vergleichbar)
- Hilfe (noch nicht programmiert, lohnt sich wahrscheinlich auch nicht)

Die Software ist bei mir verfügbar und ist für Privatpersonen Postcardware[1]. Unternehmen und Behörden können sie gegen eine kleine Gebühr[2] nutzen. Für Vorschläge in Richtung einer Erweiterung oder gar einer Zusammenarbeit[3] in der Entwicklung bin ich sehr dankbar.

[1] Der Autor möchte eine Postkarte mit einem Bild der Heimatstadt des *TEXIT!*-Users zugeschickt bekommen.
[2] Ansporn für die Weiterentwicklung.
[3] Programmierung von Unterroutinen und schwierigen Programmteilen.

Eine TEX-Version für OS/2 von DANTE e.V.?

Herr Koch schrieb in der letzten Ausgabe, daß er sich eine DANTE e.V.-eigene OS/2 TEX-Version vorstellen könnte. Ich könnte mir dies auch vorstellen und eine Offenlegung der emTEX-Sourcen von Herr Mattes für die gezielte Anpassung für OS/2 wünschen. Warum nicht eine TEX-Version für OS/2 in Verbindung mit einer graphischen Bedienoberfläche (meine *TEXIT!*-Bedienoberfläche kann dabei als Beispiel dienen) auf Basis von austauschbaren `DLL`s weiterentwickeln?

Die Beta-Version von emTEX

Die wohl zum Zeitpunkt dieses Beitrags aktuelle TEX-Version für OS/2 ist: `3c-beta12` für 386, 486 und Pentium-Rechner. Sie kann auf dem ftp/mail-Server von DANTE e.V. oder bei mir bezogen werden. Wenn die Mailbox von DANTE e.V. ihren Betrieb aufnimmt, sollte man sich die Software dort holen.

Rechtschreibreform

Luzia Dietsche

Nachdem durch die neue Rechtschreibreform das Thema „Orthographie“ wieder so aktuell wurde, fand ich es angebracht, die folgenden „fünf Schritte“ aus der Motten- und Schmunzelkiste auszukramen und zu veröffentlichen. Ich habe den Text vor längerer Zeit in einer der Diskussionslisten gesehen und mich (damals noch) köstlich amüsiert. Den Ursprung des Textes kenne ich nicht, er wurde damals auch nicht genannt. Vielleicht kann ja eines unserer Mitglieder weiterhelfen ...

Fünf Schritte zur modernen Orthographie

Erster Schritt: Wegfall der Großschreibung

einer sofortigen einführung steht nichts im weg, zumal schon grafiker und werbeleute zur kleinschreibung übergegangen sind.

zweiter schritt: wegfall der dehnungen und schärfungen

dise masname eliminirt schon di gröste felerursache in der grundschule, den sin oder unsin unserer konsonantenverdopelung hat onehin nimand kapirt.

driter schrit: v und ph werden durch f, z, tz und sch durch s ersetzt

das alfabet wird um swei buchstaben redusirt, sreibmasinen und sesmasinen fereinfachen sich, wertfole arbeitskräfte könen der wirtsaft sugefürt werden.

firter srit: q, c und ch werden durch k, j und y durch i, pf durch f ersest

iest sind son seks bukstaben ausgesaltet, di sulseit kan sofort fon neun auf swei iare ferkürst werden. anstat aksig prosent rektsreibunterikt könen nüslikere fäker wi fisik, kemi oder reknen ferstärkt gelert werden.

fünfter srit: di seiken ä, ö und ü werden durk a, o und u ersest

ales uberflusige ist iest ausgemerst. di ortografi ist wider slikt und einfak. naturlik benotigt es einige seit, bis dise fereinfakung uberal riktig ferdaut ist, fileikt sasungsweise ein bis swei iare.

anslisend durfte als nakstes sil di fereinfakung der nok swirigeren und unsinigeren gramatik anfisirt werden.

Rezensionen

TEX für Heimwerker . . .

Harald Schoppmann

Über TEX und vor allem LATEX sind bereits viele Bücher erschienen. All diesen Büchern gemeinsam ist ein anwenderorientierter Ansatz, bei dem die vielfältigen Einsatzmöglichkeiten dieses Textsatzsystems aus den unterschiedlichsten Richtungen betrachtet werden. Voraussetzung ist dabei jedoch immer ein funktionsfähiges Grundsystem von TEX. Bei der Installation und der Auswahl geeigneter Treiber ist der Systembetreuer (bzw. auf PCs der Anwender) meist ganz auf sich allein gestellt. Genau an dieser Stelle setzt das Buch „*Making TEX Work*“ von Norman Walsh an, das (nach dem Umschlagtext) ein Führer durch das Labyrinth der TEX-Tools sein will. Behandelt wird die Implementation von TEX und der im weitesten Sinne dazugehörenden Zusatzprogramme auf den gängigsten Plattformen (UNIX, MS-DOS, OS/2, Apple MacIntosh, Atari, . . .). Daneben werden die wichtigsten Makropakete kurz vorgestellt. Als Bezugsquelle für die besprochenen Pakete wird vor allem auf die CTAN-Archive (ftp/mail-Server) verwiesen, deren Struktur beschrieben wird. Kommerzielle Programme werden ebenfalls mit Herstelleradressen vorgestellt.

Das ca. 500 Seiten starke Buch besteht aus einem Vorwort, das allgemeine Hinweise über verfügbare TEX-Implementationen und Bezugsmöglichkeiten enthält, und drei Hauptteilen. Ein umfangreicher Anhang und ein ausführliches Stichwortverzeichnis runden den Text ab.

Der erste Teil erläutert die Arbeitsweise von TEX und stellt Vergleiche mit WYSIWYG-Systemen an. Das Zusammenspiel von TEX-, LATEX-, Style-, TFM- und anderer Dateien bei der Formatierung einer Eingabedatei wird anhand von Diagrammen beschrieben. Nach dieser straffen Einleitung werden als wichtigste Anwenderschnittstelle einige verbreitete Editoren und TEX-Shells besprochen. Aus der unübersehbaren Fülle von Editoren wurden diejenigen ausgewählt, für die spezielle TEX-Modes verfügbar sind. Dies sind z. B. `emacs` in Verbindung mit AUC-TEX (für UNIX/DOS) oder `epm` mit `epmtex` (OS/2). Die erwähnten TEX-Shells sind überwiegend nur für PCs verfügbar. Anschließend wird die Syntax des TEX-Aufrufs beschrieben, wobei auf systemspezifische Besonderheiten und

Fehlerquellen hingewiesen wird. Darauf folgt eine stichwortartige Vorstellung der bekanntesten Makropakete.

Der zweite Teil beschäftigt sich mit Problemen, die bei der Bearbeitung von komplexen Dokumenten entstehen. Zuerst wird erklärt, wie TEX Fonts behandelt und wie spezielle Fonts angesprochen werden können. Anschließend werden die unterschiedlichen Möglichkeiten vorgestellt, mit denen Grafiken erzeugt und in TEX-Dokumente integriert werden. Besonders ausführlich werden in diesem Zusammenhang Bildschirm- und Druckertreiber beschrieben. Daneben werden auch speziell Lösungen für mehrsprachige Texte und Online-Dokumentationssysteme behandelt. Dieser Teil wird mit einem Crash-Kurs in METAFONT und der Vorstellung von Programmen zum Generieren von Bibliographien und Stichwortverzeichnissen abgeschlossen.

Im dritten Teil werden verschiedene TEX-Implementationen vorgestellt. Besonders ausführlich wird dabei auf die freie emTEX-Verteilung für DOS und OS/2 eingegangen. Unter den kommerziellen Programmen wird vom konventionellen μTEX über Windows-Umgebungen (PCTEX for Windows) bis zum *fast* WYSIWYG-System Scientific Word alles aufgeführt, was auf TEX aufbaut. Neben dem PC wird hierbei auch der Macintosh berücksichtigt. Daran schließt sich eine lange Liste mit Kurzbeschreibungen von mehr oder weniger nützlichen Tools an, die auf den CTAN-Servern zu finden sind.

Der Anhang bildet mit über 130 Seiten eigentlich den vierten Hauptteil des Buches, der größte Teil davon sind Font-Beispiele. Daneben findet man einige nützliche Shell-Skripte und Batch-Dateien, mit denen beispielsweise automatisch Fonts durch METAFONT generiert werden können.

Ein Buch, das ein so komplexes System wie TEX und all seine Zusatzprogramme beschreiben will, kann nur eine Momentaufnahme liefern. Bedingt durch den Redaktionsschluß des Buches im Frühjahr '94 konnte deshalb auch nur die Betaversion von LaTeX2_ε besprochen werden. Im Text werden aber die unterschiedlichen Komponenten (z. B. NFSS2) behandelt. Entsprechend kann das Buch auch nicht alle Programme beschreiben, die im weitesten Sinne mit TEX zusammenhängen. Norman Walsh ist es aber gelungen, die Struktur des TEX-Systems zu beschreiben. Mit den hier vermittelten Informationen wird dem Interessierten der Weg geebnet, um Lösungen für spezielle Probleme, wie die Integration von Grafiken, das Drucken in Farbe oder die Ansteuerung eines bestimmten Druckers, zu lösen. Er liefert die nötigen Schlüssel, um den Bestand von Tools auf den CTAN-Servern zu erschließen. Die typischen Probleme bei TEX-Installationen (z. B. Pfade zu den Fontverzeichnissen) werden theoretisch

behandelt, *Kochrezepte* zur Beseitigung von Störungen sucht man dagegen vergeblich.

Das vorliegende Buch kann aber nur einem eingeschränkten Anwenderkreis empfohlen werden. Vor allem Systembetreuer von heterogenen Netzwerken finden hier vielleicht die benötigten Informationen, um Anwenderwünsche zu erfüllen. Aber auch PC-Benutzer, die Software für spezielle Aufgaben suchen und vor etwas Handarbeit bei der Installation nicht zurückschrecken, erhalten hier ebenfalls wichtige Hinweise. Der *normale Anwender*, der seine Texte mit LaTeX formatiert und ein funktionierendes TeX-System besitzt, benötigt das Buch jedoch nicht.

Das Buch ist in gut verständlichem Englisch geschrieben und eignet sich aufgrund der ausführlichen Register auch als Nachschlagewerk. Zahlreiche Bildschirmfotos, Flußdiagramme und Tabellen lockern den Text auf. Eine Fundgrube für alle, die auf der Suche nach speziellen (oder exotischen) Fonts sind, ist Anhang B, wo auf ca. 40 Seiten über 100 frei verfügbare Fonts vorgestellt werden.

Norman Walsh: Making TeX Work
O'Reilly & Associates, Inc., Sebastopol, 1994
ISBN: 1-56592-051-1
54,– DM

Spielplan

Termine

28.2.1995	LaTeX2$_\varepsilon$-Tag in Verbindung mit DANTE'95 Gießen Kontakt: Günter Partosch
1.–3.3.1995	DANTE'95 und 12. Mitgliederversammlung von DANTE e.V. Gießen Kontakt: Günter Partosch
24.–28.7.1995	TUG'95 St. Petersburg, Florida Kontakt: Mimi Burbank
4.–8.9.1995	EuroTeX'95 Arnhem, Niederlande

Stammtische

In verschiedenen Städten im Einzugsbereich von DANTE e.V. finden regelmäßig Treffen von TEX-Anwendern statt, die für jeden offen sind. Im folgenden sind die Daten und Adressen aufgelistet, die an uns weitergeleitet wurden.

12687 Berlin
Horst Szillat
Sella-Hasse-Str. 31
Tel.: 9322496 (Beantworter)
`szillat@berlin.snafu.de`
Gaststätte „Bärenschenke“
Friedrichstr. 124
Letzter Donnerstag im Monat, 19.00 Uhr

22527 Hamburg
Reinhard Zierke
Tel.: 040/54715-295
Hamb. Microcomputer-Hochschulgruppe
Grindelallee 143 (Hinterhof)
20146 Hamburg
Letzter Donnerstag im Monat, 18.00 Uhr

28759 Bremen
Martin Schröder
Tel.: 0421/628813
`l15d@alf.zfn.uni-bremen.de`
Universität Bremen, MZH 4.St.
gegenüber den Fahrstühlen
Erster Donnerstag im Monat, 18.30 Uhr

35392 Gießen
Günter Partosch
HRZ der Justus-Liebig-Universität
Heinrich-Buff-Ring 44
`guenter.partosch@`
`hrz.uni-giessen.de`
„Licher Bierstuben“, Licher Straße
Letzter Montag im Monat, 19.30 Uhr

42279 Wuppertal
Andreas Schrell
Windhövel 2
Tel.: 0202/66 68 89
`Andreas.Schrell@FernUni-Hagen.de`
Gasthaus „Yol“, Ernststr. 45
Zweiter Donnerstag im Monat, 19.30 Uhr

47226 Duisburg
Friedhelm Sowa
Rheinstr. 14
„Gatz an der Kö“, Königstraße 67
Dritter Dienstag im Monat, 19.30 Uhr

53111 Bonn
Ulrich Wisser
Am Roemerkastell 22
Tel.: 0228/692356
„Anno“, Kölnstr. 47
Dritter Montag im Monat, 20.00 Uhr

69195 Wiesbaden
Christian Kayssner
Elsässer Platz 9
Tel.: 0611/48 11 7
Andreas Klause, Elsässer Platz 3
Erster Montag im Monat, 20.00 Uhr

69008 Heidelberg
Luzia Dietsche
Tel.: 06221/29 76 6
`dante@dante.de`
China-Restaurant Palast
Lessingstr. 36
Letzter Mittwoch im Monat, 20.00 Uhr

16th Annual Meeting of TEX Users Group

I. General Invitation

The TEX Users Group is proud to announce that the *sixteenth* annual meeting will be held at the

TradeWinds Hotel, in St. Petersburg Beach
Florida, July 24 – 28, 1995

We would like to extend a warm invitation to TEX users around the world – come join us at one of the largest and most beautiful resort beaches in Florida, as we explore where TEX is to be found and how its users are going far beyond – or are diverging from – its initial mathematical context.

The theme of the meeting will be "Real World TeX" and we plan to have demonstrations of pre- and post-processors, and the active participation of developers and vendors, in hopes that *you*, the user, may discover "hands-on" just what can be done with TEX, METAFONT, PostScript, and other utilities!

Commercial users of TEX are particularly encouraged to attend. The meeting will feature papers of interest to publishers and TEX vendors, a panel discussion addressing commercial users' needs and wants, and a gallery for displaying samples of TEX work.

There will be the usual courses associated with the meeting: *Intensive Courses* in LATEX2ε and TEX, PostScript, Graphics, and perhaps other topics. The meeting itself will have excellent speakers, panel discussions, workshops, poster displays, Birds-of-a-Feather sessions (BoFs) and technical demonstrations.

II. Getting Information

A preliminary schedule will be available in February of 1995, so be sure to look for updates:

- in TUGboat and TTN (TUG publications)
- on the World Wide Web (WWW) server, at `http://www.ucc.ie/info/TeX/tug/`
- on CTAN in `tex-archive/usergrps/tug/`

In January, there will be an on-line form for registration via the WWW address noted above (the form to be called `tug95form.html`).

Send suggestions and requests to the following: `tug95c@scri.fsu.edu`.

The TUG'95 committee will assist those individuals who wish to reduce their costs by sharing accommodations. The Bursary Fund is also available to assist TEX users who demonstrate need. All members are encouraged to consider contributing to the fund. To obtain more information about contributing to or applying for the Bursary Fund, please contact the TUG Office by email to `tug@tug.org` or by post to the TEX Users Group, P.O. Box 869, Santa Barbara, CA 93102–0869 USA.

III. Deadlines

Abstracts of Papers	31 January, 1995
Preliminary Papers Due	31 March, 1995
Other Proposals[1]	30 April, 1995
Preprint Deadline	30 June, 1995
Meeting Dates	24–28 July, 1995
Camera Ready Deadline	25 August, 1995

[1] Workshops, panels, posters, demonstrations, etc.

Adressen

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

Tel.: 06221/2 97 66
Fax: 06221/16 79 06
e-mail: `dante@dante.de`

Konten: Postgiroamt Karlsruhe
BLZ 660 100 75

2134 00-757 für Beiträge
bzw. 2946 01-750 für Bücher und Disketten
bzw. 1990 66-752 für Tagungen

Präsidium:

Joachim Lammarsch	Präsident `president@dante.de`
Uwe Untermarzoner	Vizepräsident `vice-president@dante.de`
Friedhelm Sowa	Schatzmeister `treasurer@dante.de`
Luzia Dietsche	Schriftführerin `secretary@dante.de`

ftp-, mail-, gopher- und WWW-Server:

`ftp.dante.de` [129.206.100.192]	(ftp)
`ftpmail@dante.de`	(e-mail)
`gopher.dante.de`	(gopher)
`www.dante.de`	(WWW)

Autoren/Organisatoren

Mimi Burbank [42]
Supercomputer Comp. Res. Inst.
B-186, 400 Science Center Library
Florida State University
USA-Tallahasee, FL 32 306-4052
tug95c@scri.fsu.edu

Luzia Dietsche [3, 28]
siehe Seite 44

Matthias Eckermann [23]
Kaulbachstraße 29a
80539 München
Tel.: 089/2386-2868
ud311aa@sunmail.
lrz-muenchen.de

Gerd Neugebauer [4]
Ödenburger Str. 16
64295 Darmstadt
gerd@imn.th-leipzig.de

Günter Partosch [40]
HRZ der Justus-Liebig-Universität
Heinrich-Buff-Ring 44
35392 Gießen
guenter.partosch@
hrz.uni-giessen.de

Horst Peiffer [31]
Ericsson Eurolab Deutschland
Ericsson Allee 1
52134 Herzogenrath
Tel.: 02407/575364
eedhop@aachen.ericsson.se

Bernd Raichle [16]
siehe Seite 47

Dipl.-Ing. Michael Schank [34]
Egerlandweg 7
73066 Uhingen
Tel: 07161/352973

Walter Schmidt [11]
Warschauer Strasse 9/1106
99089 Erfurt
Tel.: 0361/734640

Harald Schoppmann [37]
Königsäcker 10a
68237 Schwetzingen
harald.schoppmann@
urz.uni-heidelberg.de

Uwe Untermarzoner
Kohlplattenweg 50
72074 Tübingen
Tel.: 0711/7207-4099
vice-president@dante.de

Technischer Beirat

Zuschriften an die Koordinatoren werden in der Regel nur beantwortet, wenn ein ausreichend frankierter und adressierter Rückumschlag mitgeschickt wird. Die Koordinatoren sind nicht verpflichtet, auf jede Frage einzugehen.

Amiga
Markus Erlmeier
Postfach 415
84001 Landshut
Tel.: 0871/77939
Btx: 087177939-0001
MAUS: `Markus Erlmeier@LA`
FIDO: `2:2494/106.21`
Internet: `amiga@dante.de`

Atari
Stefan Lindner
Karolinenstr. 52b
90763 Fürth
`atari@dante.de`

oder

Lutz Birkhahn
Fürtherstr. 6
90556 Cadolzburg
Tel.: 09103/2886
`atari@dante.de`

BS2000 & Graphik
Friedhelm Sowa
Heinr.-Heine Universität
Rechenzentrum
Universitätsstr. 1
40225 Düsseldorf
Tel.: 0211/3113913
`graphik@dante.de`

Macintosh
Lothar Meyer-Lerbs
Am Rüten 100
28357 Bremen
Tel.: 0421/252624
`macintosh@dante.de`

MVS
Joachim Lammarsch
Universitätsrechenzentrum
Im Neuenheimer Feld 293
69120 Heidelberg
`mvs@dante.de`

Vertreter:
Dr. Klaus Braune, s. UNIX

NOS/VE & METAFONT
Norbert Schwarz
Ruhr Universität
Rechenzentrum
Universitätsstr. 150
44721 Bochum
Tel.: 0234/700-3940
`metafont@dante.de`

PC
Dr. Peter Breitenlohner
Max-Planck-Institut für Physik
Postfach 40 12 12
80805 München
`pc@dante.de`

OS/2
Thomas Koch
Hauptstr. 367
53639 Königswinter
os2@dante.de

UNIX
Dr. Klaus Braune
Universität Karlsruhe
Rechenzentrum
Zirkel 2
76128 Karlsruhe
Tel.: 0721/608-4031
unix@dante.de

VAX/VMS
Gerhard Friesland-Köpke
Universität Hamburg
FB Informatik
Vogt-Kölln-Str. 30
22527 Hamburg
vms@dante.de

VM
Dr. Georg Bayer
TU Braunschweig
Rechenzentrum
Postfach 3329
38023 Braunschweig
vm@dante.de

German-Style
Bernd Raichle
Stettener Str. 73
73732 Esslingen
german@dante.de

Lehrerfortbildung
Werner Burkhardt
Carl-Benz-Schule Mannheim
Neckarpromenade 23
68167 Mannheim
lehrer@dante.de

PostScript
Jürgen Glöckner
In der Hessel 23
69168 Wiesloch
postscript@dante.de

Server-Koordination
Dr. Rainer Schöpf
Zentrum für Datenverarbeitung
der Universität Mainz
Anselm-Franz-von-Bentzel-
Weg 12
55099 Mainz
server@dante.de

Treiber
Joachim Schrod
Kranichweg 1
63322 Rödermark-Urberach
treiber@dante.de

Verlag und Buchhandel
Christa Loeser
Intern. Thomson Publ. GmbH
Trübnerstr. 38
69121 Heidelberg
Tel.: 06221/400177
Fax: 06221/472909
verlage@dante.de

Inhalt Heft 4/94