Die TEXnische Komödie

DANTE
Deutschsprachige
Anwendervereinigung TEX e.V.

11. Jahrgang Heft 3/1999 August 1999

3/1999

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 <1998/12/01>`, `xdvik 18f` und `windvi` (für die Bildschirmdarstellung) und `dvips(k) 5.78` (für Korrektur und Belichtung). Die Schriften zur Belichtung wurden mit dem METAFONT-Modus `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
Redaktion:	Gerd Neugebauer (verantwortlicher Redakteur)

Johannes Ammon	Jan Braun	Luzia Dietsche
Rudolf Herrmann	Uwe Münch	Thomas Nitschke
Günter Partosch	Bernd Raichle	Volker RW Schaa
Andreas Schlechte	Karin Schwind	Peter Willadt

Redaktionsschluß für Heft 4/1999: 12. Oktober 1999 ISSN 1434-5897

Editorial

Liebe Leserinnen und Leser,

in diesem Heft haben wir wieder einmal eine gute Mischung von Beiträgen der verschiedensten Art. Diese Mischung umfaßt sowohl eher Vereinsinternes, wie den Evaluationsbericht zu $\mathcal{NTS}$, als auch TEXnisches. Damit ist hoffentlich sowohl für den Einsteiger als auch den Fortgeschrittenen etwas Interessantes dabei.

Der Aufruf nach Hilfe aus dem letzten Heft wurde erhört. Gleich drei Beiträge befassen sich mit dem Thema „Interlinearübersetzung“ und präsentieren Lösungsvorschläge zu dem Problem. Dieser Erfolg sollte auch andere dazu anregen, anspruchsvolle Aufgabenstellungen in „Die TEXnische Komödie“ zu bringen. Vielleicht findet sich dann auch wieder jemand, der sich der Aufgabe annimmt und einen Lösungsvorschlag liefert.

Nicht so erfolgreich war meine Anregung zu einigen Beiträgen im vorhergehenden Editorial. Die von mir angeregten Themen sind nicht auf Gegenliebe gestoßen. Da diese Ausgabe gut gefüllt ist, zeigt das vielleicht nur, daß die Themen, die mich interessieren, bei anderen vielleicht doch nicht so im Vordergrund stehen.

In dieser Ausgabe ist auch ein Erfahrungsbericht über 4allTEX enthalten. Immer wieder bekommen die Mitglieder von DANTE e.V. CD-ROMs. Da kann es schon schwierig sein, sich für eine bestimmte Installation zu entscheiden. Ich hoffe, andere Leser nehmen diesen Erfahrungsbericht als Anregung, weitere TEX-Distributionen zu rezensieren. Damit kann dieser Erfahrungsschatz auch anderen Mitgliedern zugänglich gemacht werden, der als Entscheidungsgrundlage für den nächsten notwendigen Installationswechsel genutzt werden kann.

Das TEXnische Geschehen in DANTE e.V. findet nicht nur in dieser Zeitschrift statt. Deshalb hoffe ich, möglichst viele Mitglieder auf der nächsten Mitgliederversammlung in Heidelberg zu sehen und verbleibe bis dahin

Ihr Gerd Neugebauer

Hinter der Bühne

Vereinsinternes

Grußwort

Liebe Mitglieder,

zunächst möchten wir auf zwei Termine hinweisen: vom 20. bis zum 24. September 1999 findet in Heidelberg die TEX-Tagung EuroTEX'99 unter dem Thema „Paperless TEX“ statt. Da es schon seit längerer Zeit keine internationale TEX-Tagung in Deutschland gegeben hat, ist dies sicher eine gute Gelegenheit, über den nationalen Tellerrand hinauszuschauen.

Aus diesem Anlaß findet die Mitgliederversammlung von DANTE e.V. im Rahmen der EuroTEX statt. Gleichzeitig haben wir die Anregung aufgenommen, sie auf ein Wochenende zu legen, damit mehr Mitglieder die Möglichkeit haben, an ihr teilzunehmen. Dank des Entgegenkommens der Veranstalter der EuroTEX ist es uns möglich, die Mitgliederversammlung am Tagungsort der Konferenz abzuhalten. Sie findet am Sonntag, den 19. September 1999 statt. Näheres entnehmen Sie bitte der gesonderten Einladung.

Dieser Ausgabe von „Die TEXnische Komödie“ liegt der aktuelle Abzug des „Comprehensive TEX Archive Network“ (CTAN) auf CD-ROM bei, den wir auch dieses Jahr wieder an alle Mitglieder verteilen. Ergänzende Informationen zu dem CD-ROM-Set finden Sie ab Seite

In dieser Ausgabe findet sich auch der Evaluierungsbericht von Hans Hagen zum $\mathcal{NTS}$-Projekt (siehe Seite

Mit TEXnischen Grüßen

Thomas Koch (Präsident) Volker RW Schaa (Vizepräsident)

4allTEX-CD-ROM bei DANTE e.V.

Volker RW Schaa

Wietse Dol und Erik Frambach von der „Nederlandstalige TEX Gebruikersgroep“ (NTG) haben die fünfte Ausgabe ihrer 4allTEX-CD-ROM fertiggestellt (siehe auch `http://4tex.ntg.nl/4tex5/`). Diese Version ist ausschließlich für Benutzer von Windows 95/98/NT bestimmt, es enthält ein englischsprachiges Handbuch (550 Seiten) als PDF-Datei. Sie finden in dieser Ausgabe von „Die TEXnische Komödie“ auf Seite

Wir möchten damit die Entscheidung, sich die CD-ROM zuzulegen, jedem Interessierten erleichtern, da das Präsidium für die Verteilung des aktuellen 4allTEX-CD-ROM-Sets ein neues Verfahren beschlossen hat. In der Mitgliederbefragung, deren Auswertung im Oktober 1998 stattfand und mit „Die TEXnische Komödie“ 3/98 veröffentlicht wurde, fiel die Resonanz bezüglich dieser CD-ROM deutlich geringer aus. Eine größere Gruppe äußerte den Wunsch, die CD-ROM nur auf Anforderung zu erhalten. Deshalb möchten wir keine generelle Verteilung vornehmen, sondern nur den interessierten Mitgliedern diese CD-ROM direkt zukommen lassen.

Jedes Mitglied, das ein 4allTEX-CD-ROM-Set (Doppel-CD-ROM) wünscht, schickt einen Brief an die Geschäftsstelle von DANTE e.V. und legt einen Betrag von 3,– DM in Briefmarken bei. Von der Geschäftsstelle wird dann das CD-ROM-Set an das Mitglied versandt.

The $\mathcal{NTS}$ project (p)reviewed
April 22, 1999
Revised: May 15, 1999

Hans Hagen

Introduction

On April 16 and 17, the $\mathcal{NTS}$ team met in Brno in the Czech Republic. The $\mathcal{NTS}$ project team consists of representatives from the TEX community and a

professional programmer. This programmer, as well as a considerable part of the expenses involved in administering and running the project, are paid by the German speaking TEX users group DANTE e.V. Although the initial estimate was one full-time year of work by the programmer (Karel Skoupý), it became clear during the first year of development that an additional year of funding would be needed. On behalf of DANTE e.V., Hans Hagen was asked to attend the NTS meeting and to report on the progress in relation to the additional funding. Although in principle funding was guaranteed, it was felt necessary to get an impression in what way the money was being spent, and how much money would be needed to finish this part of the project.

This report will be limited to a rather general description. There will be more detailed reports from the NTS team concerning some of the technical issues as discussed at the meeting.

Objectives

The long term objective of the NTS project is to provide the TEX community with an extensible environment for typographical programming. A first step in the development of this environment is to provide a full functioning alternative for traditional TEX, rewritten in such a way that extensions can easily be implemented and experiments with new typesetting methods can easily be conducted. This first version of NTS will be referred to as NTS version 0.

The first version of NTS will be TEX compatible, which means that it has to pass the so-called trip test. Because this test uses the log file output, it means that not only the typographical output must be identical (this can be tested by comparing DVI files) but it also means that, apart from system and time dependent features, everything not typographic must behave exactly the same.

Methods

The programming language used is Java. This object oriented language is available on many platforms and is highly standardized when it comes to distributing both source and compiled code.

TEX has been written by Donald Knuth, using Pascal. Because Pascal in its standardized form lacks modularization, the WEB system was developed first. This system enabled Knuth to organize the source code in such a way that the program becomes sort of a story, to be read from start to end. This way of dealing with programming is called literate programming. If we consider

typesetting to be the primary objective of TEX, the extensive documentation as well as individual components of the program itself clearly shows the secondary objective: *educational*.

From the well-documented source code, the final TEX Pascal code is generated, compiled and converted to its platform dependent binary form. Currently NTS cannot be considered a literate programming effort; the programming conventions used are derived from Java programming practice (JavaDoc). Because the NTS team has taken upon itself the task to deliver a well documented, readable and extensible program, quite some effort still has to go into commenting the source code.

At the moment TEX emerged, computers were not yet very fast. They had limited memory, and efficient storage of data was an important issue. Differences in architecture meant that portability had to be guaranteed by clever programming.

As a result, some of the internals of *TEX The Program* that were necessary at that time may be considered obsolete when modern programming environments are used. NTS version 0 has to be trip-test compatible. This means that some of the obsolete low-level features still have to be implemented, even if they oppose the idea of a New Typesetting System. Although it is quite clear that to be successful, NTS has to be TEX compatible to start with, building-in these *hacks* takes time. The TEX way of dealing with characters in the range 128–255, for instance (using the `^^` conventions), results in methods that clearly need additional explanations, if only to make sure that those extending the code understand why some of those hacks are there.

There has been some discussion within the team as to how to deal with the naming of constants and variables. Is it still feasible to talk about `leftbrace` in situations where `bgroup` might be more appropriate? Similar decisions must be made for `hlist` and `vlist`. Should these become `linelist` and `pagelist`? For the moment it's left to the programmer to make the right choice, although some such decisions are reviewed during group meetings. Generalization must not harm the understanding of the processes taking place in TEX.

Depending on the aspect at hand, extending can be fairly straightforward. Many hard-coded limitations and initializations are isolated in such a way that they no longer pose a limit. Registers are implemented using a hash table, and removing the 256 *allocation limit* does not take any additional programming. Instead of using a string pool, NTS follows a rather more resource-hungry method, but compensates for this *greed* by providing a more suitable framework for localization. We were warned that overloading and extending functionality

can harm the integrity of the program. This will pose some demands on the documentation.

Results

Roughly 50% of the 1300 modules that make up *TEX The Program* have been re-implemented in Java. A substantial amount of the programmer's time was spent on becoming familiar with the Java programming environment and the internals of TEX. A clear estimation of how much time the actual programming takes is hard to give. The existing TEX data structures and functionality are mapped onto classes and methods, and the hierarchy of classes is evolving rapidly. Currently there are about 200 classes in 100 files, mostly dealing with mode-independent issues. The classes are organized in packages called `base`, `io`, `lang`, `font`, and `tex`. At the moment, cyclic dependencies are eliminated, though we were warned that they may creep in before the task is complete. There are some thoughts about configuration files that may be used for initialization, turning on extensions, and alike.

Until recently, the programmer has spent $\frac{1}{3}:\frac{1}{3}:\frac{1}{3}$ of the time to learning, designing and programming. The amount of time spent on learning is currently decreasing, although it is unlikely ever to reach zero.

As said previously, *TEX The Program* is more than only lines of code: it is an example of literate programming. The principles behind the algorithms as well as the typographic issues involved are explained in detail in *TEX The Program*, as written by Knuth. Unless new functionality is added, which will not happen in this very first stage, documentation in NTS can be limited to explaining the code. Although anyone can adapt the code, experiences with the original TEX show that only a limited group of people will probably do so. Therefore the documentation can be provided at the level of a moderately experienced programmer who is familiar with the nature of NTS. Currently about 20% of the documentation needed is finished, and this mainly concerns interfacing issues. This documentation is done in the Java way, conforming to the JavaDoc specifications. This means that 80% of the more detailed documentation of the currently finished code is still to be done.

More extensive documentation of the whole code will be provided, although when possible, references to the original code will be made. Given that it should be done *in the spirit of Knuth*, this task should not be underestimated. It is definitely a NTS group activity, but if needed arrangements could be made for others (perhaps computer science students with a good grasp of the requirements of literate programming) to do part of the job.

To enable proper and clear documentation, which in itself can have an educational side effect, *hacks* should be avoided. For this reason there will be a strong pressure on using pure Java techniques instead of dirty tricks (e.g. set constructors instead of bit masks).

Planning

At the immediately preceding review (in Dortmund, Germany), the additional programming effort required was estimated to be about two months. This rather optimistic view was proved to be wrong. Not only did it take more time to unearth the many features of TEX, but reorganizing (introducing classes) and coding took much more time than expected. Even though this is due to the fact that re-implementing can take more time than starting from scratch, one may wonder why a more realistic planning had been so difficult. The NTS team has learned a lesson and has made a more realistic, detailed and in some respects more conservative planning this time. A careful analysis of the process afterwards can probably provide more insight in the conversion of an entangled program written in a more or less linear programming language (one big source) into an object oriented alternative split up in many parts.

Most of the work yet to be done concerns the parts that deal with typesetting and producing DVI output. The NTS group has made a plan for the packages (collections of classes) still to be implemented. The next table lists the work to be done; the third column lists the number of weeks provided for programming. Integration, discussion, documentation and testing and debugging the full NTS program is not part of this planning.

	package(s)	*week(s)*
1	extensions / reading and writing files	1
2	conditionals	1
3–5	boxes and lists / packaging / shipping out	5
6	building pages	3
7	paragraph building and hyphenation	5
8	dumping / undumping / initialization	1
9	alignment	4
10	math	6

At EuroTEX'99, a demonstration will be given of the basic typesetting functionality. Maths typesetting as well as that of alignments will be implemented

just after EuroTEX. A proposal for a presentation at EuroTEX has been prepared, and most members of the group will participate in this presentation. A demonstration will be given on two platforms; actually showing some results was considered to be of more importance than trying to squeeze maths and alignments into $\mathcal{NTS}$ without proper testing.

Most isolated functionality is tested in the process, but once maths and alignments are in place, testing of the full functionality can take place. This will be around December 1999/January 2000. At that moment, or perhaps even earlier, a major decision must be made concerning incorporation of functionality present in the current successors. There is no doubt that version 1 should offer ε-TEX functionality, while pdfTEX makes a good candidate for version 2.

Currently TEX is not only the program (binary) TEX, but also a collection of programs that deal with the handling of font data (metrics and glyphs), hyphenation patterns, and DVI files. Of course, there are also METAFONT and METAPOST. In all of these programs, similar or nearly similar code fragments can be found. Because in an object-oriented environment such as Java, code re-use is effectively guaranteed, the results of the first stage are more than TEX alone. Many of the programs that form a part of the TEX suite will automatically be in place, as soon as their underlying functionality is also present in TEX. Some TEX implementations call TEX-related programs from within TEX, for example to generate and convert fonts. In the end, this will be handled in $\mathcal{NTS}$ in a more natural way, since the underlying code (classes and methods) is actually the same.

The official release of $\mathcal{NTS}$ version 0 will take place at TUG 2000 in the UK. The $\mathcal{NTS}$ group thinks it is best not to come up with incomplete intermediate distributions. It is in the spirit of TEX to do things properly and to offer only stable and documented versions. Distribution will be eased by the Java feature of zipped file collections.

As soon as a stable pre-release exists, it will be made available to a limited group of testers.

Meetings

To guard progress as well as to prepare for EuroTEX, the $\mathcal{NTS}$ group has scheduled a meeting for July 10/11. In addition, the group will continue to communicate by e-mail, and Karel will continue to upload new versions of the code to the access-restricted $\mathcal{NTS}$ server. The next meeting after that will take place directly during or just after EuroTEX.

Funding

When the implementation of NTS started, and funding was provided, re-implementation was estimated to take about one year. This optimistic view was probably triggered by the advocated qualities of Java and the availability of a well-documented original TEX. For various reasons, implementation proper actually started some months after the official starting point. The new planning sets the moment of completion to somewhere at the beginning of 2000, but testing and documentation will quite certainly take some more time. Therefore this first stage can best be set to take 2.5 years, of which about one-half has already passed.

Given the importance and use of extensions such as ε-TEX, pdfTEX and alike, it makes sense to seek funding to ensure that NTS version 1 (i.e. ε-NTS) will be available soon after version 0 has been made public. It is also important to consider what should be done with METAFONT and METAPOST, being the graphical relatives to the mainly text-based TEX.

The NTS team

The project team (at present) consists of a managing director, a technical director, a project leader, and two members. For personal as well as practical reasons the managing director stepped down to become a member, while the project leader took over as managing director. At the meeting the official team was defined as:

Name	*former application*	*new application*
Joachim Lammarsch	managing director	
Jiří Zlatuška	project leader	managing director
Philip Taylor	technical director	technical director
Peter Breitenlohner		
Friedhelm Sowa	treasurer	treasurer
Karel Skoupý	programmer	programmer

The programmer plays a central role in the redesign of the system, so the term *programmer* should be interpreted rather liberally.

In the process of reorganization, I was asked if I was willing to join the team and take the role of project leader. Because it would complicate my task as observer, I expressed the willingness to join and take responsibilities, but the definitive answer to this request will be given when this report is accepted and

continuity guaranteed. I think it does no harm to identify the responsibilities associated with the three roles mentioned.

Conclusions

What conclusions can we draw? Looking at the objectives, we can conclude that re-implementation is undoubtedly possible. The $\mathcal{NTS}$ team demonstrated that TEX can be split up into a number of meaningful classes and methods in such a way that extensions are possible. Although the main typesetting part is still to be programmed, there is no reason to believe that it can not be achieved in the proper way.

A second conclusion is that the difficulty of re-implementing TEX has been underestimated. Among the possible explanations, the requirement to be trip-test-compatible pays its toll. On the other hand, compatibility is a necessity, since otherwise $\mathcal{NTS}$ will not be accepted as a valid successor to TEX.

Thirdly, I conclude that because the next stage will concern the main features of TEX, and one of the focus points of extensions (typesetting), greater participation by the $\mathcal{NTS}$ team members will be expected (by the programmer) as well as needed (on behalf of planning). The project could benefit from more reporting and discussion on the dedicated $\mathcal{NTS}$ mailing list. The proposed detailed time schedule also gives the project team more opportunities to be aware of progress and/or difficulties.

As a result of the modified schedule, the fourth conclusion is that additional funding will be needed. $\mathcal{NTS}$ will only be a good starting point for future development if the functionality provided by ε-TEX, pdfTEX, and alike are incorporated. Therefore a minimum of two years of additional funding must be sought. DANTE e.V. is willing to provide part of this funding, but it is hoped that other user groups will participate too. Given the international character of the project, it makes sense to look for funding by (for instance) the European Community.

Remark

The meeting showed that it is not always easy to find the right balance between *getting a clear picture* and *setting the next steps*. Although the programmer is highly competent and experienced in object-oriented programming, the project can, and in my opinion will, benefit from a bit more control and involvement in the programming effort. The $\mathcal{NTS}$ project must be anchored firmly in the

TEX community and the presentation and meeting at EuroTEX, as well as the intermediate meeting in July are therefore very important.

Personally I think this project is very important for the TEX world. (Because the project identifies what is needed when converting a non object-oriented program to an object-oriented one, it can even serve a purpose outside the TEX community.) There is not much doubt that by mid-2000 there will be a stable and portable New Typesetting System version 0 and maybe even version 1. In my opinion, the TEX community should also take its responsibility after those releases. Stability, consistency and quality has provided TEX its place in the world, and NTS should be in the same spirit.

> At first I thought it would be easy; I expected that the job could be done in a few months. [...] Boy, was I wrong! All my life I have underestimated the difficulty of the projects I've embarked on, but this was a new personal record for being too optimistic.
>
> Donald E. Knuth, Digital Typography, October 1998

Bretter, die die Welt bedeuten

LaTeX: Das ideale Satzsystem für blinde Studierende in naturwissenschaftlichen Disziplinen?

Erdmuthe Meyer zu Bexten, Jens Hiltner

Einleitung und Problemdarstellung

In der Bundesrepublik Deutschland gibt es über 155 000 blinde Menschen, wobei 16 Prozent davon jünger als 40 Jahre alt sind. Darüber hinaus gibt es noch eine viel größere Anzahl von stark Sehbehinderten. Diese beläuft sich auf über 550 000 Menschen in Deutschland (mit steigender Tendenz). Im Vergleich noch zu einigen Jahren zuvor hat sich den Sehbehinderten und Blinden eine ganz neue Welt im Berufsleben eröffnet. Das klassische Berufsfeld als Telefonist und Masseur wird um neue, viel attraktivere Berufe erweitert. Heutzutage haben sie auch die Möglichkeiten, ein Studium durchzuführen. Es stehen ihnen die verschiedensten Studiengänge wie beispielsweise Jura, Journalistik, Informatik, Betriebswissenschaft oder Germanistik offen. Dieser Fortschritt ist darin begründet, daß immer leistungsfähigere und auch kostengünstigere Computer mit zugehöriger spezifischer Soft- und Hardware sowie andere moderne Informationstechniken entwickelt wurden, die gerade den Sehgeschädigten enorme Möglichkeiten und Hilfen bieten. Offiziell im Studium zugelassen wurden Computer aber erst im Jahre 1996.

Besonders in den naturwissenschaftlichen und technischen Studiengängen spielt die Mathematik eine bedeutende Rolle. In den Vorlesungen (wie beispielsweise Physik, Elektrotechnik oder Chemie) werden die Studierenden immer wieder mit (mathematischen) Formeln konfrontiert, was ihnen verschiedene Probleme bereitet. Gleiche Probleme existieren aber auch in der Ausbildung beispielsweise zum Fachinformatiker oder Verwaltungsangestellten, wo Mathematik ein Lehrfach darstellt.

Was kann dagegen getan werden? Es gibt verschiedene Lösungsansätze, die im Rahmen dieses Beitrages diskutiert werden sollen. Schon seit vielen Jahren wurden immer wieder Computer-Mathematikschriften (wie z. B. [1, 2]) speziell für Blinde konzipiert. Mit Hilfe dieser besonderen Schriftsysteme wird zum einen versucht, Blinden das Schreiben von mathematischen Formeln über die normale Computer-Tastatur zu ermöglichen und zum anderen die Formeln für sie auf der Braille-Zeile[1] auszugeben und damit lesbar zu machen. Später wurde das System auf ein 8-Punkt-Raster erweitert, so daß nun 256 Zeichen codierbar sind. Im neuen Zentrum für Blinde und Sehbehinderte an der Fachhochschule Gießen-Friedberg werden für sehgeschädigte Studierende spezielle Unterstützungen bei der Durchführung ihres Studiums in den Fachbereichen Mathematik, Naturwissenschaften und Informatik (MNI) und Wirtschaft (W) angeboten. An der Hochschule wird keine der verschiedenen existierenden Computer-Mathematikschriften eingesetzt. Wir verfolgen eine ganz andere Richtung, indem wir das weltweit anerkannte und verbreitete Satzprogramm LaTeX [3] einsetzen.

Dies beruht auf folgenden Tatsachen: Derzeit gibt es in der Bundesrepublik Deutschland leider keine einheitliche und verbindliche Computer-Mathematikschrift für den Schulbereich. Da unsere Studierenden aus den unterschiedlichsten Schulen im Bundesgebiet kommen, haben sie dementsprechend jeweils verschiedene Mathematikschriften kennengelernt. Weiterhin kommt erschwerend hinzu, daß die Schriftsysteme untereinander nicht kompatibel sind, was für den Austausch und die Kommunikation untereinander von großem Nachteil ist. Auf Grund dieser fehlenden Orientierung und der Unverbindlichkeit kommt als weiteres Problem hinzu, daß mehr und mehr „Privatschriften“ erfunden werden (vgl. [4]). Bei Bedarf werden im Unterricht spontan neue mathematische Symbole erfunden. Darüber hinaus sind die Mathematikschriften auch nicht unbedingt sehr benutzerfreundlich, gerade auch im Bezug auf Sehende, d. h. ein Austausch zwischen Sehenden und Blinden ist kaum möglich. Alle diese zuvor genannten Gründe haben uns deshalb veranlaßt, unseren blinden Studierenden das System LaTeX zu lehren.

Studium an der Fachhochschule Gießen-Friedberg

An der Fachhochschule Gießen-Friedberg wurde vor einigen Monaten, genauer gesagt am 4. Dezember 1998, ein neues Zentrum für Mensch-Maschine-Kommunikation insbesondere für Blinde und Sehbehinderte (kurz BliZ genannt)

[1] Die Braille-Schrift wurde Ende des 19. Jahrhunderts (1885) von dem damals 16jährigen blinden Franzosen Louis Braille entwickelt. Hierbei handelt es sich um ein tastbares Schriftsystem für Blinde. Sie basiert auf einem 6-Punkt-Raster, d. h. es können damit $2^6 = 64$ verschiedene Zeichen (und auch ganze Wörter) kodiert werden.

eröffnet, das den sehgeschädigten Studierenden in den Fachbereichen Mathematik, Naturwissenschaften und Informatik (MNI) und Wirtschaft (W) eine auf ihre Behinderung optimal ausgerichtete Unterstützung für ihr Studium bietet und somit erheblich zum Nachteilsausgleich gegenüber ihren nicht behinderten Kommilitonen beitragen soll. Damit ist die Fachhochschule Gießen-Friedberg in der Bundesrepublik Deutschland Vorreiter für die sehbehinderten- und blindengerechte Ausbildung zum Informatiker oder Betriebswirt.

Das Besondere am BliZ besteht darin, daß es nicht einfach eine Vielzahl technischer Hilfsmittel in Form von Hard- und Software zur Verfügung stellt, sondern eine integrierte Arbeitsumgebung für die Studierenden bietet. Hierzu gehören neben Computer-Arbeitsräumen auch ein Besprechungsraum und eine Teeküche. Diese Einrichtung bietet den äußeren Rahmen für eine qualifizierte Ausbildung in Berufsfeldern, die Sehgeschädigten bislang kaum an Fachhochschulen zugänglich waren. Darüber hinaus werden im Rahmen von Projekten in diesem neuen Zentrum Hilfsmittel, Software usw. für Sehgeschädigte getestet, weiter- oder neuentwickelt.

Die technische Ausstattung ist sehr umfangreich. Es stehen verschiedene Rechner mit in Größe und Art unterschiedlichen Monitoren sowie zwei Notebooks für das Arbeiten unterwegs oder für Übungen und Praktika zur Verfügung. An die Rechner und Notebooks sind unterschiedliche Braille-Ausgabezeilen angeschlossen. Weiterhin gehören zu der Ausstattung zwei Bildschirmlesegeräte, zwei unterschiedliche Scanner und verschiedene Drucker (Laser- und Farb-/Tintenstrahldrucker sowie ein Braille- und ein Taktildrucker). Zum Austausch von Daten und Informationen steht eine komplexe Kommunikationseinrichtung vom Telefon über Fax bis hin zu mehreren Videokonferenzsystemen zur Verfügung. Darüber hinaus gibt es zur Dokumentation von Arbeitsabläufen und zum Training von Gesprächs- und Vortragssituationen eine komplette Video-Ausrüstung.

Neben dieser Hardware gibt es im BliZ unterschiedliche Software-Systeme für die Sprachein- und -ausgabe, Texterkennung und Bildvergrößerung für verschiedene Betriebssysteme (Windows NT/95/98, OS/2 und Linux). In der Zentralbibliothek wurde ein spezieller Bereich für die blinden- und sehbehinderten-spezifischen Werke – also für Bücher, CD-ROMs und Hörkassetten – eingerichtet.

Über Fernleihe kann auf weitere blinden- und sehbehinderten-gerechte Literatur von anderen Bibliotheken in der Bundesrepublik zurückgegriffen werden. Außerdem stellen einige Verlage ihre Werke direkt auf CD-ROM oder Diskette, d. h. in digitaler Form, zur Verfügung.

Mathematikschriften

In den nachfolgenden Unterabschnitten sollen kurz einige übliche Mathematikschriften für Blinde vorgestellt werden, wie

- Marburger Mathematikschrift
- Karlsruher und Dresdner ASCII-Mathematikschrift (AMS)
- Stuttgarter Mathematikschrift (SMSB)
- Bochumer Mathematik-Punktschrift (BMPS)

Ausführliche Informationen über die jeweiligen Schriften können in [4] nachgelesen werden, oder es kann dazu Literatur direkt von den verschiedenen Einrichtungen angefordert werden. Die Mathematikschriften haben alle eine entsprechende Darstellung im Braille-Zeichensatz und dazu als ASCII-Zeichen auf dem Bildschirm.

Marburger Mathematikschrift

Die Marburger Mathematikschrift wurde im Jahre 1955 von blinden Mathematikern für blinde Mathematiker entwickelt. Allerdings spielte zur damaligen Zeit die Darstellung und das Arbeiten mit dem Computer noch keine Rolle.

Andere Kriterien (siehe [4]) wie Übersichtlichkeit, Handhabbarkeit, Kompaktheit bzgl. Platzausnutzung des Punktschriftpapiers und Kompatibilität zur traditionellen „Literatur-Punktschrift“ standen dafür im Mittelpunkt.

Karlsruher und Dresdner ASCII-Mathematikschrift (AMS)

Zur Darstellung mathematischer Formeln in wissenschaftlichen Texten wird an den Universitäten in Karlsruhe und Dresden die „ASCII-Mathematikschrift für Blinde“ (AMS) verwendet. Diese Notation wurde seinerzeit am Modellversuch „Informatik für Blinde und hochgradig Sehbehinderte“ der Universität Karlsruhe entwickelt und vom Studienzentrum für Sehgeschädigte der Universität Karlsruhe und der Arbeitsgruppe „Studium für Blinde und Sehgeschädigte“ der Technischen Universität Dresden überarbeitet. Üblicherweise erfolgt die Darstellung mathematischer Symbole am Computer in der Regel grafisch, was allerdings für Blinde und hochgradig Sehbehinderte so nicht zu lesen ist. Um mathematische Symbole zugänglich zu machen, werden sie in eine Zeichendarstellung (AMS) konvertiert. Der Zeichenvorrat ist hierbei auf den auf Computern verfügbaren ASCII-Zeichensatz beschränkt. Damit sind die erstellten

Texte systemunabhängig und können mit jedem einfachen Editor gelesen und verändert werden. Mit der AMS wird das Ziel verfolgt, neben syntaxorientierten Schreibweisen für mathematische Ausdrücke auch solche einzuführen, die die Semantik der Ausdrücke besser hervorheben. Die AMS erlaubt außerdem, für häufig vorkommende Ausdrücke verkürzte Schreibweisen zu verwenden. Beides trägt zur Verbesserung der Lesbarkeit bei. Weitere und detailliertere Informationen können im Internet unter [5] abgerufen werden.

Stuttgarter Mathematikschrift (SMSB)

Das Hauptziel der von Frau Dr. Schweikhardt (Universität Stuttgart, Fachbereich Informatik) 1981 entwickelten und 1989 überarbeiteten Stuttgarter Mathematik-Schrift lag darin, daß eine eindeutige 1:1-Zuordnung von Punkt- und Schwarzschriftzeichen ermöglicht werden sollte. Nur so kann eine optimale Kommunikation zwischen Sehenden und Blinden gewährleistet werden.

Bochumer Mathematik-Punktschrift (BMPS)

Im Rahmen des Projektes „Integration blinder und hochgradig sehbehinderter Schüler in der zuständigen Regelschule“ am Heinrich-von-Kleist-Gymnasium (Bochum) wurde eine weitere Mathematikschrift entwickelt. Der Erfinder dieser Schrift ist Herr Jandrik Kraeft [7]. Sein Ziel war, eine Schrift speziell für Schülerinnen und Schüler zu entwickeln, in der auch die Struktur der Ausdrücke deutlich werden sollte. Beispielsweise werden bei einem Bruch die Ziffern im Zähler anders markiert als diejenigen, die im Nenner stehen. Leider ist diese Schrift für Sehende nicht leicht zugänglich und die Darstellung auf dem Computer für Nichtblinde schwer verständlich.

MATHS

Eine ganz andere Richtung verfolgt MATHS (Mathematical Access for Technology and Science for Visually Disabled Users) [8], um sehbehinderten und blinden Menschen den Zugang zur Mathematik zu erleichtern. Ziel des Projektes MATHS ist es, den interaktiven Umgang blinder und sehbehinderter Schülern und Studenten mit Mathematik durch einen interaktiven PC-basierten Arbeitsplatz zu verbessern. Im Rahmen des MATHS-Projektes wird die sogenannte MATHS-Workstation entwickelt. Diese soll einem blinden oder sehbehinderten Benutzer mathematische Formeln mittels akustischer Ausgabe (Sprache und Töne) und Braille-Schrift vermitteln. Die akustische Ausgabe soll einen allgemeinen Überblick über die mathematische Struktur der Formel ausdrücken.

Eine genaue Darstellung kann durch Sprache und Braille-Ausgabe gewonnen werden. Die Eingabe von Texten und mathematischen Ausdrücken erfolgt mittels der Tastatur, Sprache und Braille-Zeile. Analysen ergaben, daß neben der Eingabe von mathematischer Blindenschrift oder der Eingabe per gesprochener natürlicher Sprache auch Eingabemöglichkeiten zur elementaren Umformung der Gleichungen notwendig sind (beispielsweise Bewegen eines Terms auf die andere Gleichungsseite). Der eigentliche Editor zum Lesen, Erstellen und Verändern mathematischer Dokumente ist ein auf MS-Windows basierender SGML-Editor mit WYSIWYG-Darstellung. Die interne Repräsentation basiert auf der EUROMATH-DTD und weicht damit von der für HTML 3.0 ursprünglich vorgeschlagenen DTD stark ab.

Die syntaktischen Strukturen der EUROMATH-DTD können auch nach LaTeX konvertiert werden. Der SGML-Editor wurde mit einer DDE-Schnittstelle versehen, so daß die interne SGML-basierte Repräsentation für die Braille-Darstellung oder akustische Ausgaben verwendet werden kann. Mathematiktexte und -formeln werden automatisch vorgelesen und die korrekte Prosodie[2] berücksichtigt (derzeit für Englisch und Flämisch).

Um einen Überblick über den Aufbau einer möglicherweise komplexen Formel zu erhalten, wurde ein Übersichtsmodus entwickelt (audio glance), der den einzelnen Termarten MIDI-Musikinstrumente zuordnet. Damit kann eine Verkürzung um ca. 60 Prozent im Vergleich zur verbalen Ausgabe erreicht werden.

Neben der akustischen Ausgabe wird auch die Braille-Ausgabe unterstützt. Sowohl die Ausgabe für flämische Mathematikschrift als auch für die Stuttgarter Mathematikschrift ist bereits realisiert worden. Die interaktiven Möglichkeiten sind dabei besser als mit akustischer Ausgabe, da Zeigehandlungen auf modernen Braille-Anzeigen möglich sind und somit die Direktheit der Interaktion erhöht wird. Dieses Projekt befindet sich noch in der Weiterentwicklung. Zusätzliche Informationen können bei der Firma Papenmeier in Schwerte angefordert werden.

LaTeX

Das Satzprogramm LaTeX [3] beruht auf dem von Donald E. Knuth (Standford University, USA) Mitte der 70er Jahre entwickelten Satzsystem TeX. Nach ersten brauchbaren Ergebnissen, die 1978 erzielt wurden, begann seine kontinuierliche Verbreitung. Seit Mitte der 80er Jahre hat TeX eine weltweite

[2] Lehre von den Tonhöhen und der Quantität der Silben

Verbreitung gefunden; das heißt es ist inzwischen nahezu auf jedem Rechnertyp und jedem Betriebssystem verfügbar. Ein Grund dafür liegt wohl darin, daß der Programmautor Donald E. Knuth es zum öffentlichen Eigentum (public domain) erklärt hat. Der Nachteil des sehr leistungsfähigen TeX liegt aber darin, daß gerade dessen Anwendung und insbesondere die Ausschöpfung aller Möglichkeiten erhebliche Erfahrungen mit der Programmierung voraussetzen. Das bedeutet aber auch, daß dadurch dieses System nicht allen Menschen zur Verfügung stand.

Dieses große Manko brachte den amerikanischen Computer-Wissenschaftler Leslie Lamport [6] darauf, das Programmpaket LaTeX zu entwickeln, das auf TeX aufbaut. Dafür ist aber LaTeX viel anwendungsfreundlicher geworden, da es auch ohne große Programmierkenntnisse zu nutzen ist (aufgrund der viel benutzerfreundlicheren Schnittstelle zwischen TeX und dem Anwender). Da auf TeX basierend, ist es auf allen Betriebssystemen verfügbar. Ein wesentlicher Grund für die Entwicklung von LaTeX ist auch darin begründet, daß gerade im Zeitalter des Computers nicht nur die Blinden, sondern auch die Sehenden das Problem hatten, mathematische Formeln bzw. Texte über den Computer einzugeben. Auf der Tastatur befinden sich nämlich nicht alle notwendigen mathematischen Symbole und Zeichen, um mathematische Ausdrücke zu beschreiben. Abgesehen davon kann auch die mathematische Struktur der Ausdrücke, d. h. deren Aufbau, nicht gewährleistet werden. Eine weitere Besonderheit von LaTeX besteht darin, daß die Notation auf Textzeichen des 7-Bit-ASCII-Codes basiert, was bedeutet, daß sie sowohl in Schwarzschrift als auch in Punktschrift eindeutig präsentiert werden kann.

LaTeX vs. Mathematik-Schriften

Die im vorherigen Kapitel kurz vorgestellten Mathematikschriften sind ganz speziell entwickelte Schriftsysteme für Blinde mit dem Ziel, ihnen die mathematischen Formeln „sichtbar“ (fühlbar) zu machen. LaTeX hat gegenüber den Mathematik-Schriften verschiedene Vorteile, die im folgenden aufgeführt und kurz erläutert werden:

1. Ein ganz großer Vorteil von LaTeX für Blinde ist darin zu sehen, daß es ein weltweit bekannter, verbreiterter und anerkannter Standard ist und dementsprechend auch gerade von vielen Nicht-Sehbehinderten intensiv genutzt wird. Darüber hinaus ist im wissenschaftlichen Bereich eine große Verbreitung zu verzeichnen. Denn egal, für welche Mathematikschrift man sich auch entscheidet, stellt diese immer eine Insellösung dar und ist nicht unbedingt von Sehenden leicht zu interpretieren. Ganz zu schweigen davon, daß

es sehr schwierig sein wird, Sehende dazu zu bewegen, eine dieser speziellen Schriften zu erlernen.

2. LaTeX kann zum einen als Textformatierungssystem und zum anderen zur Darstellung und Beschreibung mathematischer Formeln und Tabellen eingesetzt werden.

3. Die Beschreibung komplexer Tabellen und mathematischer Formeln erfolgt textbasiert und zeilenorientiert, so daß eine Ausgabe und damit Kontrolle über eine Braille-Ausgabezeile immer leicht möglich ist.

4. LaTeX ist im Gegensatz zu MS-Word kein WYSIWYG-System, da die Formatierung in Textform beschrieben wird. Das führt dazu, daß sich die Lesbarkeit von korrekt formatierten Dokumenten gerade für Blinde über die Braille-Zeile viel einfacher und komfortabler gestaltet.

5. Ein Besonderheit bei LaTeX ist, daß Dokumentklassen (Styles) für verschiedene Zielformate – beispielsweise Briefe (`letter`), Berichte (`report`), Bücher (`book`) – existieren, die speziell von Setzern ausgearbeitet wurden und nun jedermann zur Verfügung stehen. Darüber hinaus können auch eigene Dokumentklassen definiert werden (siehe beispielsweise die Klassen vom Springer-Verlag oder aber die Klasse `dtk` für diesen Artikel). Die Vorgabe von Dokumentklassen hat für Blinde einen großen Vorteil in dem Sinne, daß sie sich nicht mehr um die sorgfältige Formatierung ihrer Texte kümmern müssen.

   Dieser Aspekt mag sich zuerst trivial anhören, ist jedoch für Menschen, die nicht sehen können, wie ihr Dokument aussieht, von großer Bedeutung. Durch die Verwendung vorgegebener Formatierungen, etwa für Überschriften, Literaturstellen, Vorworte, usw., ist sichergestellt, daß das Aussehen des Textes auch ohne die Kontrolle so ist, wie man selbst es erwartet.

6. Das System LaTeX ist auf praktisch allen Computer- und Betriebssystemen verfügbar.

7. Für Neulinge in diesem Bereich gibt es zahlreiche Literatur von Einführungsbüchern bis hin zur Spezialliteratur (siehe beispielsweise [3, 6, 9, 10]).

8. Im Zeitalter der elektronischen Medien, die bereits verstärkt Einzug in die Lehre gefunden haben, spielt auch LaTeX eine große Rolle. Skripten und Übungszettel werden häufig in LaTeX erstellt und über Netz an die Studierenden weitergereicht [11].

9. Während die vorgestellten Mathematikschriften aus einer Fülle von mathematischen Schreibweisen für verschiedene mathematische Inhalte bestehen,

bietet LaTeX ein einfaches Konzept an, das sich in wenigen Regeln zusammenfassen läßt (vgl. [4]).

10. Die Kürzel für spezielle mathematische Symbole sind in LaTeX nach allgemeinverständlichen, mnemonischen Gesichtspunkten zusammengestellt (vgl. [4]).

11. Die Präfixnotation, d. h. das Voranstellen der Befehlsworte – wie `\frac` oder `\sqrt` – vor den jeweiligen Ausdruck erleichtert die Lesbarkeit für den sequentiell lesenden Blinden (vgl. [4]).

12. Es gibt eine Fülle von Mathematikliteratur, die mit LaTeX geschrieben wurde (vgl. [4]). Gleiches gilt auch für verschiedene Computer-Bücher, wie zum Beispiel [12, 13].

Vergleich der Systeme

Im folgenden erfolgt ein kurzer Vergleich der in den vorherigen Abschnitten vorgestellten Computer-Mathematik-Schriften mit LaTeX. Die in den Tabellen 1–3 aufgeführten Beispiele wurden aus [4] übernommen.

Meinungen von blinden Studenten

Wie bereits im vorherigen Abschnitten erwähnt, wird im BliZ derzeit LaTeX anstelle einer Mathematikschrift eingesetzt. Im folgenden stellen die blinden Studenten, die derzeit im BliZ studieren und LaTeX hier an der Hochschule gelernt haben, ihre Meinung zu LaTeX vor.

Zusammenfassend berichteten die Studierenden:

> „Grafische Benutzeroberflächen und Anwendungen sind in den letzten Jahren die Regel geworden. So zeigt es sich, daß auch die Erwartung von z. B. Professoren an den abgelieferten Schriftstücken wächst (mit einem Computer erstellt, Aussehen eines gedruckten Werkes). Eine Schreibmaschine konnte diesen Standard nie erreichen, selbst einfache Textverarbeitungsprogramme unter DOS leisteten dies nicht. Was jedoch für sehende Menschen selbstverständlich und zum Alltag werden kann, ist für Blinde und Sehbehinderte ein großes Problem. Wie soll denn auch eine blinde Person einen Text gestalten können, wenn sie das Layout nie zu sehen bekommt? Da können auch Hilfsmittel nicht helfen, welche zwar über den Inhalt Aufschluß geben, nicht aber über Formatierung und Farben. Auch im Bereich der Darstellung mathematischer

Table 1: Vergleich der Bruchschreibweise

Schriftsystem	*Schreibweise*
Symbolik für Sehende	$\frac{(a+b)^{n+1}-c}{a-b}$
Stuttgarter Schrift	`▌(a+b)⫫ n+1╫ -c▬a-b▌`
Karlsruher Schrift	`((a+b)**(n+1)-c)/(a-b)`
LaTeX	`\frac{(a+b)^{n+1}-c}{a-b}`

Table 2: Vergleich der Wurzelschreibweise am Beispiel der p-q-Formel

Schriftsystem	*Schreibweise*
Symbolik für Sehende	$x_{1,2}=-\frac{p}{2}\pm\sqrt{\frac{p^2}{4}-q}$
Stuttgarter Schrift	`x╖1,2╫ = -p/2 ± √p ╒ 2/4 -q ╫`
Karlsruher Schrift	`x(1,2;) = -p/2 + //(p**2/4 -q)`
LaTeX	`x_{1,2}=-\frac{p}{2}\pm\sqrt{\frac{p^2}{4}-q}`

Table 3: Vergleich der Integralschreibweise

Schriftsystem	*Schreibweise*
Symbolik für Sehende	$\int_0^x \sqrt{1-t^2}dt$
Stuttgarter Schrift	`0⌠x √1-t╒ 2╫ dt`
Karlsruher Schrift	`Int[0;x] //(1-t**2)dt`
LaTeX	`\int_0^x\sqrt{1-t^2}dt`

Zusammenhänge wird das Problem deutlich, insbesondere bei grafischen Sonderzeichen, die von den Hilfsmitteln gar nicht oder nur teilweise dargestellt werden. Gehen wir jedoch davon aus, daß ein Textdokument in irgendeiner Form einmal Klartext gewesen sein könnte, scheint eine Lösung in Sicht: LaTeX.

Wie auch von HTML-Dokumenten bekannt, haben wir einen Klartext, welcher mit Befehlen versehen und somit elektronisch weiterverarbeitet werden kann. Es werden quasi Routinen, die bei Windows-Anwendungen nicht sichtbar sind, einfach ausgeschrieben, wie beispielsweise das Zentrieren von Texten. Auch Skalierung und Schriftartenwechsel sind möglich, wobei hier ausschließlich ein Standard-Texteditor verwendet werden kann. Durch LaTeX wird blinden und sehbehinderten Menschen ermöglicht, komplexe Textdokumente zu erstellen, ohne sie zuvor gesehen zu haben.

Auch im Bereich der Mathematikschrift zeigt sich, daß LaTeX eine willkommene Lösung ist. Formeln werden nämlich ebenfalls in Klartext dargestellt und lassen sich Zeichen für Zeichen erfassen. Warum ist nun ausgerechnet LaTeX interessant und nicht die von anderen Universitäten entwickelten Blinden-Mathematikschriften? Es gibt zwar zahlreiche Systeme, jedoch sind alle unterschiedlich und bedürfen eines aufwendigen Lernprozesses.

LaTeX ist mittlerweile ein Standard geworden und eine Kompatibilität zwischen Blinden und Sehenden ist gewährleistet. Mit Konvertern und anderen Hilfsmitteln kann aus LaTeX-Dokumenten ein ebenso „schöner Text“ erstellt werden, wie z. B. mit Microsoft-Word. Ferner bietet LaTeX einen großen Vorteil: Es ist kostenlos, während WinWord ca. 600 DM kostet.

Warum ist LaTeX als Mathematikschrift für Blinde interessant? Neben LaTeX gibt es mehrere Varianten der Mathematikschriften. Wie beispielsweise bei der Marburger Punktschrift (siehe Seite 17) wäre es nur schwer möglich, diese Schrift auf einem Computer darzustellen, da die Zeichen optisch keinen Sinn ergeben würden. Aus diesem Grund ist diese Schrift für den Umgang mit modernen Informationstechniken zwar eher ungeeignet, für den schulischen Einsatz wohl die beste und einfachste Lösung und kann parallel zur Blindenkurzschrift erlernt und genutzt werden. Eine andere Variante der Mathematikschrift wird an der Technischen Universität in Dresden eingesetzt.

Diese Schrift wird aus logischen Sonderzeichen generiert, die z. B. für Spezialfunktionen doppelt geschrieben werden. Sehende Menschen, die diese Schrift lesen können, haben am Bildschirm keine Probleme bei der Erfassung. Beide Systeme haben jedoch den Nachteil, daß sie in Hinblick auf die große weite Welt Alleingänge darstellen und zu Lernzwecken sicher Anwendung finden können, jedoch die Kommunikation zu Außenstehenden nicht ermöglichen.

LaTeX vereint alle Vorteile obiger Systeme miteinander und ermöglicht zusätzlich die Erstellung professioneller Texte. Ferner ist LaTeX mittlerweile ein Standard, so daß auch Außenstehende mit LaTeX-Dokumenten von Blinden arbeiten könnten. Zwar ist LaTeX aufgrund seiner Komplexität eher schon eine eigene Programmiersprache, die sicher mehr Aufwand zum Erlernen erfordert. Der Ertrag ist jedoch höher, weil Blinde und Sehbehinderte sicher mehr Nutzen von LaTeX haben als von einer bloßen Mathematikschrift."

Abschlußbemerkungen

Bereits im ersten Jahr des BliZ haben sich die Vorteile von LaTeX gegenüber den anderen Mathematikschriften gezeigt. Obwohl das Erlernen von LaTeX für die Blinden nicht einfach war, überzeugten die Vorteile, da es auch von nichtsehbehinderten Kommilitonen eingesetzt wird, was die Zusammenarbeit und somit die Integration der Studenten erheblich fördert, ja erst gar ermöglicht. Von der Ausdrucksstärke der Systeme sind eigentlich alle in der Lage, beliebig komplexe Formeln zu beschreiben, jedoch bietet LaTeX weitaus mehr als nur eine weitere „Mathematikschrift"; auf Grund der Dokumentklassen und der umfangreichen Möglichkeiten für den Textsatz handelt es sich um ein universelles Werkzeug. Unter diesen Gesichtspunkten gilt hier die Forderung, LaTeX möglichst frühzeitig schon in der Schule als Beschreibungssprache zu lehren, anstatt eine nur „lokal bekannte" Eigenentwicklung zu unterrichten.

Danksagungen

An dieser Stelle möchten sich die Autoren bei Herrn Helmut Kopka nochmals recht herzlich für die Ermunterung zur Erstellung dieses Beitrages bedanken. Ebenso bedanken wir uns bei Herrn Günter Partosch für Anregungen und Korrekturen zu diesem Artikel. Darüber hinaus möchten wir uns bei den blinden Studierenden Herrn S. Merk, Herrn H. Oschwald und Herrn H. Sahin vom Fachbereich Mathematik, Naturwissenschaften und Informatik der Fachhochschule

Gießen-Friedberg für ihre Unterstützung bei der Erstellung dieses Beitrages bedanken.

References

[1] *Marburger Mathematikschrift*, Deutsche Blindenstudienanstalt (BLISTA), Marburg, 1998.

[2] *ASCII-Mathematikschrift (AMS) für Blinde*, Universität Karlsruhe, Studienzentrum für Sehgeschädigte (SZS), Karlsruhe, Mai 1994.

[3] Helmut Kopka: *LaTeX – Eine Einführung*, Addison-Wesley Longman, Bonn, 1997.

[4] Ulrich Kalina: *Welche Mathematikschrift für Blinde soll in den Schulen benutzt werden?*, Beiheft Nr. 5 der Zeitschrift blinde/sehbehinderte, Zeitschrift für Sehgeschädigten-Pädagogik, Heft 3, S. 78–94, 1998.

[5] Informationen zur *Mathematikschrift für Blinde*, `http://elvis.inf.tu-dresden.de/asc2html/ams/h-000001.htm#1`.

[6] Leslie Lamport: *Das LaTeX Handbuch* Addison-Wesley Longman, Bonn, 1995.

[7] Jandrik Kraeft: *Vorschlag für eine 6/8-Punkt-Mathematikschrift*, blinde/sehbehinderte, Zeitschrift für Sehgeschädigten-Pädagogik, Heft 1, 1998.

[8] Informationen zu *MATHS*, `http://www.informatik.uni-stuttgart.de/ifi/ds/MATHS.html`.

[9] Helmut Kopka: *LaTeX – Ergänzungen – Mit einer Einführung in METAFONT*, Addison-Wesley Longman, Bonn, 1997.

[10] Helmut Kopka: *LaTeX – Erweiterungen*, Addison-Wesley Longman, Bonn, 1997.

[11] Erdmuthe Meyer zu Bexten, Axel Schumann-Luck: *Elektronische Medien in der wissenschaftlichen Ausbildung für sehgeschädigte Studierende – Neue Perspektiven für blinde und sehbehinderte Studierende?!*, Fachtagung: Elektronische Medien in der wissenschaftlichen Weiterbildung, Braunschweig, 18.–19. März 1999.

[12] Michael Kofler: *Linux*, Addison-Wesley Longman, München, 1999.

[13] Michael Kofler: *VBA-Programmierung mit Excel 2000*, Addison-Wesley Longman, München, 1999.

Interlinearübersetzung: eine Möglichkeit

Christopher Creutzig

In *Die TEXnische Komödie 2/1999* [**?**] stellte Siegfried Splett das Problem, eine einfach zu verwendende Umgebung für *Interlinearübersetzungen* zu programmieren. Mit einer kleinen Einschränkung stellt dieser Artikel das Gewünschte zur Verfügung.

Gesucht war Folgendes[1]: Aus einer Eingabe der Art

```
\begin{interlinear}
  Asked, & Befragt, & at which hour & um welche Stunde &
  one should eat, & zu speisen sei, &
  Diogenes said, & sagte Diogenes: &
...
\end{interlinear}
```

sollte folgende Ausgabe erzeugt werden:

Asked,	at which hour	one should eat,	Diogenes said,	“If you are ri
Befragt,	um welche Stunde	zu speisen sei,	sagte Diogenes:	„Wenn Du reic

eat,	when you want,	if you are poor,	when you have.”
iss,	wann Du willst,	wenn Du arm bist,	wenn Du hast.“

Hierzu definiert das Paket `interlin` zwei interne Befehle und die Umgebung `interlinear`. Der erste der beiden internen Befehle, `\call@interlinear`, prüft mit Hilfe von `\@ifnextchar`, ob als nächstes Token ein `\end` folgt. Der Befehl `\@ifnextchar` vergleicht, auch wenn der Name etwas anderes suggeriert, Tokens, nicht Zeichen. Ist das der Fall, so nimmt das Paket stark vereinfachend an, daß die aktuelle `interlinear`-Umgebung beendet wird, wenn

[1] Im Originalartikel war als Beispiel eine Übersetzung polnisch/deutsch angegeben. Da ich der polnischen Sprache nicht mächtig bin, fällt es mir leichter, den Text stattdessen in Englisch wiederzugeben. Der Name der Umgebung ist bei mir nicht `doppel`, sondern `interlinear`, um international verständlich zu sein.

nicht, wird angenommen, daß weiterer Text folgt. Um diesen zu setzen, ruft `\call@interlinear` den zweiten internen Befehl auf, `\interline@r`. Dieser Befehl mußte per `\def` definiert werden, da `\newcommand` keine Möglichkeit vorsieht, daß Argumente durch besondere Trennzeichen wie beispielsweise ein `&` getrennt sind. Um die Vorteile von `\newcommand` dennoch nutzen zu können, wird der Befehl zunächst per `\newcommand` leer angelegt und diese Definition anschließend mit `\def` überschrieben.

Die Definition `\def\interline@r#1&#2&{...}` besagt, daß `\interline@r` zwei Argumente erwartet, die jeweils durch ein `&` terminiert sind. Das erledigt zwar die Aufgabe, die zusammengehörigen Textteile als Argumente zu erhalten, führt aber auf der anderen Seite dazu, daß eine fehlerhafte Eingabe wie z. B.

```
\begin{interlinear}
  This is wrong. & Dies ist falsch.
\end{interlinear}
```

ohne ein schließendes `&` eine kryptische Fehlermeldung

```
Runaway argument?
 wenn Du hast."' \end {interlinear}
! Paragraph ended before \interline@r was complete.
<to be read again>
                   \par
l.16

?
```

zur Folge hat. Eine bessere Fehlerbehandlung wäre jedoch deutlich aufwendiger.

Nachdem `\interline@r` seine Argumente gelesen hat, setzt es sie einfach in eine `tabular`-Umgebung, wobei der zuerst eingegebene Text oben in der Standardschrift erscheint und der zweite Text darunter in `\textsf`. Wiederum ist dies eine einfache Möglichkeit, die vielleicht noch ergänzt werden sollte, so daß es über eine Option bei `\usepackage` oder auch ein Argument der `interlinear`-Umgebung möglich wäre, die Auswahl der Schriften und die Positionierung der Textteile zueinander zu wählen. Vermutlich verlangt die typische Anwendung aber nur nach genau einer konsistenten Wahl, und diese kann im Paket eingestellt werden. Danach setzt es noch ein Leerzeichen als potentiellen Punkt für einen Zeilenumbruch.

Die Umgebung `interlinear` schließlich setzt den Zeilenabstand auf das 2,5-fache des normalen Wertes, so daß insgesamt ein halber Zeilenabstand zwischen

den Zeilen eingefügt wird, und ruft `\raggedright` auf, da TEX innerhalb von `tabular`, die von `\interline@r` gesetzt werden, keinen Zeilenumbruch vornehmen kann. Anschließend wird `\call@interlinear` gestartet. Am Ende der Umgebung wird mit `\par` ein Absatzende eingefügt, damit der Text auch mit den lokalen Einstellungen gesetzt wird.

Insgesamt sieht das Paket so aus:

```
%
% interlin.sty
%
% ... Kommentare
%
\NeedsTeXFormat{LaTeX2e}
\ProvidesPackage{interlin}%
  [1999/05/29 v0.1 Christopher Creutzig]

\newcommand{\interline@r}{}
\def\interline@r#1&#2&{%
  \begin{tabular}[b]{c}#1\\%
    \textsf{#2}\end{tabular}%
  \ %
  \call@interlinear
}

\newcommand{\call@interlinear}{%
  \@ifnextchar\end{}{\interline@r}%
  }

\newenvironment{interlinear}{%
  \raggedright
  \baselineskip=2.5\baselineskip
  \call@interlinear
}{\par}
```

Interlinearübersetzung: ein Vorschlag

Klaus Lagally

In Heft 2/1999 von *Die TEXnische Komödie* fragt Siegfried Splett nach einer bequemen Möglichkeit, in zweisprachigen Texten Satzteile einander zuzuordnen und jeweils zweizeilig untereinander abzusetzen [**?**]. Hier ist ein Lösungsvorschlag.

Einführung

Wir haben zwei Probleme zu lösen: ein bequemes Eingabeformat festzulegen, und dann die Formatierung zu realisieren. Dazu sehen wir uns das Beispiel aus [**?**] genauer an (wir lassen die polnischen Sonderzeichen weg; Kundige mögen dies verzeihen).

Pytajacemu sie o ktorej godzinie obiadowac nalezy, rzekl Diogenes:
Befragt um welche Stunde zu speisen nötig sei, sagte Diogenes:

„Jeslis bogaty, jedz, kiedy ci sie chce, jeslis ubogi – kiedy masz.“
„Wenn du bist reich, iss, wann du willst, wenn du bist arm – wann du hast.“

Das Ganze sieht fast aus wie ein normaler Textabsatz, aber jedes „Wort“ ist nun ein zweizeiliger Block. Die Zeilenabstände sind entsprechend größer, und weil die „Wörter“ nicht sinnvoll getrennt werden können, brauchen wir Flattersatz. Außerdem kann der Schriftstil in den einander jeweils zugeordneten Teilen unterschiedlich sein.

Schnittstelle

Für die bequeme Eingabe stellen wir zur Verfügung:

- eine Umgebung `Zeilen`, die das Absatzformat bestimmt und sinnvolle Voreinstellungen vorgibt; diese können noch lokal geändert werden,
- zwei Umgebungen `Zeilea` und `Zeileb`, die jeweils den Stil der beiden Teilzeilen vorgeben,
- ein Makro `\Paar` für die Eingabe zweier zugeordneter Satzteile.

Jedes Argument von `\Paar` bildet eine Gruppe für sich.

Damit sieht der Eingabetext für unser Beispiel nun folgendermaßen aus:

```
\renewenvironment{Zeileb}
{\sffamily} {}

\begin{Zeilen}
\Paar {Pytajacemu sie}    {Befragt}
\Paar {o ktorej godzinie} {um welche Stunde}
\Paar {obiadowac}         {zu speisen}
\Paar {nalezy,}           {n"otig sei,}
\Paar {rzekl Diogenes:}   {sagte Diogenes:}
\Paar {"‘Jeslis bogaty,}  {"‘Wenn du bist reich,}
\Paar {jedz,}             {iss,}
\Paar {kiedy ci sie chce,}{wann du willst,}
\Paar {jeslis ubogi --}   {wenn du bist arm --}
\Paar {kiedy masz."’}     {wann du hast."’}
\end{Zeilen}
```

Hier haben wir den Stil der jeweils zweiten Zeile umgestellt, aber für die erste Zeile die Voreinstellung übernommen.

Erste Realisierung

Die Realisierung der drei Umgebungen ist sehr einfach:

```
\newenvironment {Zeilen}
{\par \rightskip 0pt plus 1fil \parindent 0pt \lineskip 5mm }
{\par}

\newenvironment {Zeilea} {}{}
\newenvironment {Zeileb} {}{}
```

Die Voreinstellung für `Zeilea` und `Zeileb` ist leer.

Die nächstliegende Realisierung von `\Paar` ist, die beiden Argumente je in eine eigene Box zu packen und dann zentriert untereinander auszugeben. Dazu reservieren wir zwei Boxen:

```
\newsavebox {\intboxa}
\newsavebox {\intboxb}
```

lesen die beiden Argumente ein und packen sie zusammen:

```
\newcommand {\Paar}[2]{% Argumente lesen
\sbox \intboxa {\begin{Zeilea}#1\end{Zeilea}}%
\sbox \intboxb {\begin{Zeileb}#2\end{Zeileb}}%
\Paarx }
```

Wir balancieren die beiden Boxen, packen ihren Inhalt jeweils zentriert in eine weitere Box passender Breite, und geben die neue Box als „Wort“ aus.

```
\newcommand {\Paarx} {\dimen0 \wd\intboxa \dimen2 \wd\intboxb
\ifdim \dimen0 < \dimen2 \dimen0 \dimen2 \fi
\leavevmode \vtop {\hsize\dimen0 \parfillskip 0pt
\leftskip 0pt plus 1fil \rightskip 0pt plus 1fil
\unhbox \intboxa \break \unhbox \intboxb }}
```

Das funktioniert bereits in den meisten Fällen, hat aber noch Nachteile: die Argumente werden jetzt so früh gelesen, daß manche Umstellungen (etwa von aktiven Zeichen, oder andere Änderungen der ‘catcodes’ in den Umgebungen `\Zeilea` und `\Zeileb`) nicht mehr wirksam werden.

Verbesserung

Die Grundidee der Verbesserung besteht darin, die beiden Argumente nicht mehr als Parameter einzulesen, sondern jeweils in der passenden Umgebung auszuführen. Die Details sind recht trickreich, und daher verlassen wir jetzt den eingeschränkten Bereich der LaTeX-Programmierung und gehen auf die Grundfunktionen von TeX zurück, einschließlich einiger Kunstgriffe aus dem berühmten Anhang D im TeXbook [?].

Wir öffnen die erste Box mit `\bgroup`, stellen die lokale Umgebung mit `\Zeilea` ein, öffnen eine weitere Gruppe, hinter der es mit `\Paara` weitergeht, nehmen die führende Klammer des ersten Arguments mit `\let\next=` weg und führen es anschließend aus:

```
\newcommand {\Paar} {% jetzt keine Parameter!
\setbox\intboxa \hbox\bgroup \Zeilea
\bgroup \aftergroup\Paara \let\next=}
```

Die schließende Klammer des Arguments hat die neue Gruppe beendet; mit dem vorgemerkten Kommando `\Paara` geht es weiter. Wir schließen die Box und die Umgebung mit `\endZeilea \egroup`, und versuchen denselben Trick mit dem zweiten Argument.

```
\def \Paara {\endZeilea \egroup \futurelet\next \Paarb }
```

Hierbei müssen wir aber vorher den Fall abfangen, daß die nächsten Tokens vor dem zweiten Argument Blanks sind, die wir explizit wegnehmen müssen. Dazu sehen wir uns mittels `\futurelet` das nächste Token an.

```
\def \Paarb {\ifcat\space \noexpand\next \let\next \Paarc
\else \let\next \Paard \fi \next }
```

Um ein Blank wegzunehmen, bauen wir uns ein Makro zusammen, das ein explizites Blank im Parametermuster enthält. Das geht so:

```
\expandafter \def \expandafter \Paarc \space
{\futurelet\next \Paarb }
```

Hier haben wir das zweite Argument erreicht, und können es ausführen:

```
\def \Paard {\setbox\intboxb \hbox\bgroup \Zeileb
\bgroup \aftergroup\Paare \let\next=}

\def \Paare {\endZeileb \egroup \Paarx }
```

Jetzt haben wir auch die zweite Box mit dem zweiten Argument gefüllt, und es geht weiter wie in unserer ersten Lösung.

Hinweise zur Benutzung

Daß das Ganze funktioniert, zeigt dieser Beitrag selbst: das Beispiel oben ist mit diesen Makros formatiert.

Weil die beiden aktuellen Argumente von `\Paar` *nicht* als Parameter gelesen, sondern ausgeführt werden, dürfen sie auch lokale Deklarationen, Änderungen der ‘catcodes’, ja sogar `\verb`-Einschübe enthalten. Andererseits ist `\Paar` samt seinen Argumenten fragil und darf nicht im Argument weiterer Makros vorkommen, und auch nicht im Ersetzungstext, außer als allerletztes Token.

Interlinear-Versionen: ein Vorschlag

Peter Schmitt

In Heft 2/99 von *Die TEXnische Komödie* wurde von Siegfried Splett als „Aufgabe“ die Frage gestellt, wie in (LA)TEX Interlinear-Versionen von Texten, wie sie für zweisprachige Ausgaben manchmal verwendet werden, realisiert werden können [**?**].

Ich möchte dieses Thema im folgenden aufgreifen, dabei aber nicht sofort eine Lösung vorstellen und erklären, sondern Schritt für Schritt beschreiben, wie sie erarbeitet werden kann. Ich verwende dazu TEX-Primitive, weil das universeller, durchsichtigier und effizienter als die Beschränkung auf LATEX ist.

Vorbemerkungen

Schon nach kurzem Nachdenken stellt man fest, daß das Problem, wie oft in ähnlichen Fällen, vielschichtiger ist als es auf den ersten Blick zu sein scheint. Denn während das (oberflächlich betrachtet: Haupt-)Problem, das Finden einer entsprechenden Codierung in TEX, recht leicht zu lösen ist, bleiben zwei „Rand“-Fragen notwendigerweise ohne eine eindeutige Antwort:

Wie soll das Ergebnis, also der gesetzte Text, eigentlich aussehen?

Und: Wie soll die Eingabe, also die Syntax, gestaltet werden?

Denn: Die Anworten auf diese Fragen hängen (natürlich!) vor allem von der Art des geplanten Einsatzes ab, und damit auch von den Vorlieben und Abneigungen des Autors/Anwenders/Benutzers.

Die Grundidee

Wenden wir uns also zunächst einmal dem TEXnischen Problem zu: Es geht darum, jeweils zwei kurze Text-Bestandteile übereinanderzustellen. Natürlich können diese dazu, wie schon bei der Problemstellung angedeutet wurde, in zwei `\hbox`-en gepackt und in einer `\vbox` übereinander positioniert werden. Aber einfacher – vor allem, wenn die beiden Zeilen zentriert werden sollen – ist es, sie als Teile einer zweizeiligen und einspaltigen Tabelle aufzufassen, also im Prinzip so vorzugehen:

```
\halign { \hfil #\hfil   \cr
            Haupttext    \cr
```

```
          paralleler Text \cr
        }
```

Ein Makro, das ein Textpaar Original–Übersetzung setzt, und in der Syntax LaTeX-Makros wie `\frac` entspricht, könnte also zum Beispiel so aussehen:

```
\def\interlinearpair #1#2{%
      \vbox {%
    \halign {\hfil ##\unskip \hfil \cr
                   #1\cr
  \it\ignorespaces #2\cr
           }}%
                        }
```

`\unskip` macht das Makro unempfindlich gegen Leerzeichen am Ende des Arguments. Leerzeichen zu Beginn der Eintragung in ein Tabellenfeld werden zwar von `\halign` nicht beachtet, da aber das zweite Argument erst nach der Fontangabe eingesetzt wird, muß davor ein `\ignorespaces` verwendet werden.

Damit ist allerdings noch nicht alles getan, denn so ein Paar ist ja nur ein Baustein von vielen, die noch zu Zeilen angeordnet werden müssen. Da innerhalb der Textpaare ein Zeilenumbruch wohl nicht erwünscht und bei Verwendung von `\halign` auch gar nicht möglich ist, wird man wohl auf Randausgleich verzichten und entweder `\raggedright` verwenden oder die Zeilen zentrieren. Außerdem wird man vermutlich zwischen den Paaren etwas mehr Abstand lassen als den `\spaceskip` zwischen den Bestandteilen eines Paares. (Vielleicht wird man ihm sogar eine gewisse Dehnbarkeit (`glue`) zusprechen.) Ebenso wird man wahrscheinlich zur besseren optischen Gliederung den Abstand zwischen den Zeilenpaaren etwas größer wählen als den Abstand zwischen den Zeilen eines Paares.

Um diese Merkmale festzulegen, ist die Angabe einiger Parameter erforderlich, die ich, zur bequemen Handhabung, (zumindest vorläufig, vgl. die abschließenden Bemerkungen) mithilfe eines Makros zusammenfasse. Die konkreten Zahlenwerten dienen nur als Beispiel, mit ihnen kann und soll natürlich noch experimentiert werden.

```
\def\ILlayout {%
     \leftskip  0pt plus 1fil              % zentrieren
     \leftskip  0pt                        % linksbündig
    \rightskip  0pt plus 1fil
    \parindent  0pt
```

```
   \parfillskip  0pt
\interpairskip  10pt
                     }
\newskip\interpairskip
\def\ILspace {\hskip\interpairskip }
```

Für den Abstand zwischen den Paaren wurde ein Skip-Register verwendet, da dies in der Anwendung effizienter ist, weil die Glue-Angabe nicht jedesmal neu gelesen werden muß. Sollte aber, etwa unter LaTeX, kein freies Register mehr zur Verfügung stehen, kann statt der Zuweisung in `\ILlayout` natürlich auch die folgende Definition verwendet werden.

```
\def\ILspace  {\hskip10pt}
```

Um den Abstand zwischen den Zeilenpaaren anzugeben, könnte man den Wert von `\baselineskip` entsprechend ändern. Dann müßte man ihn aber innerhalb jedes Interlinear-Paares wieder zurücksetzen, da sonst auch zwischen den beiden Texten eines Paares derselbe große Abstand verwendet würde. Daher verwende ich ein passendes `\strut`, d. h. eine (unsichtbare) Box mit passend gewählter vertikaler Ausdehnung, mit deren Hilfe eine größere Tiefe (`depth`) der Zeilen vorgetäuscht wird:

```
\newbox\ILstrutbox
\setbox\ILstrutbox =
       \hbox{\vrule height 14pt depth 8pt width 0pt}
\def\ILstrut {\unhcopy\ILstrutbox }
```

Das Markup

Nach diesen Vorbereitungen können wir uns dem zweiten Teil der Aufgabe zuwenden und eine geeignete Notation entwickeln.

Da es unpraktisch ist, jedes Paar explizit anzugeben, wird man eine Umgebung definieren, die dies automatisch erledigt. Aber wie soll diese aussehen? Siegfried Splett schlägt vor, `&` als (einziges) Trennsymbol zu verwenden. Mir erscheint es logischer und übersichtlicher zwei Trennsymbole – eines vor der ersten Sprache, ein anderes vor der zweiten Sprache – zu verwenden. Ich werde aber beide Möglichkeiten besprechen.

Ein Ansatz zur Verwirklichung, vermutlich sogar der naheliegendste, ist, ein Hilfsmakro zu verwenden, das die Argumente als „delimited argument“ aufnimmt, an `\interlinearpair` weiterreicht und dabei noch den waagrechten und den senkrechten Zwischenraum anfügt:

```
\def\makeinterlinearpair #1&#2&{%
    \interlinearpair {#1}{#2}\ILstrut\ILspace
                              }
```

Diese Vorgangsweise hat den Nachteil, daß die Textteile als Argumente eines Makros eingelesen werden, sodaß nachträgliche `\catcode`-Änderungen unterbunden werden, was bei Verwendung von „exotischen“ nicht-englischen Schriften unter Umständen zu Schwierigkeiten führen kann. Außerdem müßte noch, unter Prüfung, ob überhaupt ein weiteres Paar vorhanden ist, für einen rekursiven Aufruf von `\makeinterlinearpair` gesorgt werden. Insgesamt ein recht großer Aufwand, der aber glücklicherweise gar nicht notwendig ist!

Denn da die Textteile ohnehin in der für die Verarbeitung benötigten Reihenfolge angegeben werden, kann die Verwendung von Argumenten leicht vermieden werden: Man muß nur durch geeignete Makros die entsprechenden Teile des `\halign`-Codes zwischen den Bestandteilen des Textpaares einfügen:

```
\def\beginpair {\vbox \bgroup \halign \bgroup
                \hfil ##\unskip \hfil \cr }
\def\midpair   {\cr \it \ignorespaces }
\def\endpair   {\cr \egroup \egroup }
```

Angenehmer Nebeneffekt: Diese Methode ist auch effizienter, denn die Argumente müssen nur einmal gelesen werden. Mit diesen Makros sieht die Eingabe eines einzelnen Interlinear-Paars nun so aus:

```
\beginpair erste Sprache \midpair zweite Sprache \endpair
```

Sie ergibt:

erste Sprache
zweite Sprache

Wenn man genau hinschaut, fällt auf, daß die untere Zeile nicht präzise zentriert zu sein scheint. Dieser Eindruck wird durch die Neigung der Italics hervorgerufen. Wenn man will, kann man versuchen, korrigierend einzugreifen, und die Textteile ein wenig nach links rücken:

```
\def\midpair  {\cr \it \kern-1pt\ignorespaces }
```

Nach dieser Modifikation sieht das Paar so aus:

erste Sprache
zweite Sprache

Eine Möglichkeit, dieses Markup kürzer, aber gleichzeitig auch möglichst einfach und übersichtlich zu gestalten, ist die zeilenweise Eingabe in zwei Spalten:

```
\bIL      erste Sprache = zweite Sprache
            eine Phrase = die übertragene Phrase
     eine andere Phrase = und deren Übersetzung   \eIL
```

Wir benötigen dazu zunächst einen Befehl, der eine für Interlineartexte geeignete Umgebung einrichtet: `\bIL` (beginInterLinear)

```
\def\bIL {\begingroup   % alle Änderungen bleiben lokal
          \ILlayout     % Umschalten auf Interlinear-Layout
          \ILmarkup     % Einschalten des Interlinear-MarkUp
          \leavevmode   % Umschalten auf horizontalen Modus
          \beginpair    % es folgt das erste Paar
          }
```

Bei der vorgeschlagenen Notation braucht jedes `=` bloß ein `\midpair` zu bewirken, während an den Zeilenenden mehr geschehen muß: Das vorangegangene Paar muß beendet und ein neues begonnen werden, außerdem ist noch für den richtigen Abstand zu sorgen. Dazu dient das folgende Makro:

```
\def\interpairs {\endpair          % beende vorangehendes Paar
                 \ILstrut          % strut anhängen
                 \ILspace          % Abstand einfügen
                 \beginpair        % beginne nachfolgendes Paar
                }
```

Nun muß noch dafür gesorgt werden, daß `=` und `^^M` (Carriage Return) wie gewünscht interpretiert werden, was mit aktiven Zeichen leicht zu bewerkstelligen ist, auch wenn das in Standard-LaTeX nicht allzu gern gesehen wird.

Die entsprechende Zuweisung wird zu Beginn der Umgebung durch das schon vorsorglich vorbereitete `\ILmarkup` vorgenommen und bleibt durch Verwendung einer Gruppe auf die Interlinear-Umgebung beschränkt, damit keine Konflikte mit anderen Paketen auftreten können.

```
\begingroup     % = und cr müssen während der folgenden
                % Definitionen lokal aktiv sein
  \catcode`\=\active
  \catcode`\^^M\active
  \global\def\ILmarkup       % global: wir sind in einer Gruppe!
        {\let   =\midpair    %  = steht in der Mitte eines Paares
```

```
          \let ^^M\interpairs% cr steht zwischen Paaren
     \catcode  `\=\active    %  = aktivieren
     \catcode`\^^M\active    % cr aktivieren
         }
\endgroup       % = und ^^M werden wieder zurückgesetzt
```

Schließlich fehlt noch `\eIL` (endInterLine), das die Interlinear-Umgebung beendet:

```
\def\eIL {\endpair  % beende das letzte Paar
          \ILstrut  % strut - vielleicht steht
                    %   das letzte Paar alleine in einer Zeile
          \par      % beende den Paragraphen,
                    %   solange die Parameter noch gültig sind
          \endgroup % Rückkehr zu normalem Text
         }
```

Das folgende Zitat (nach dem TEXbook) von George Bernard Shaw zeigt die Interlinear-Umgebung in Aktion:

When a proof has been sent me with two or three lines
Wenn eine Druckfahne wurde geschickt mir mit zwei oder drei Zeilen

so widely spaced as to make a grey band
so weit auseinandergezogen daß sie bewirken ein graues Band

across the page, I have often rewritten the passage so as to fill up
über die Seite ich habe oft die Stelle umgeschrieben sodaß sie ausfüllt

the lines better;
die Zeilen besser;

Ein alternatives Markup

Nun zurück zu der ursprünglichen Aufgabe. Auch die von Siegfried Splett vorgeschlagene Eingabe

```
\bIL erste Sprache & zweite Sprache & nochmals
     erste & und zweite Sprache \eIL
```

läßt sich auf ähnliche Weise leicht verwirklichen, man muß nur dafür sorgen, daß `&` seine Bedeutung zwischen `\midpair` und `\interpairs` wechselt: Dazu verwenden wir:

```
\begingroup \catcode`\&\active
  \gdef\Midpair   {\unskip\global\let &\Interpairs \midpair }
             % \interpairs, es folgt \midpair
             % global, weil sie innerhalb einer Gruppe
             % (gebildet durch \vbox,\halign) verwendet wird
  \gdef\Interpairs {\unskip
       \global\let &\Midpair \interpairs }
             % \interpairs, es folgt \midpair
             % diese Zuweisung erfolgt ebenfalls \global,
             % da man globale nicht mit lokalen Zuweisungen
             % mischen soll
```

Durch die `\let`-Anweisungen wird das `\unskip` aus dem `\halign` „versteckt“, daher muß es hier explizit angegeben werden. Will man (nur) die zweite Version benutzen, so kann man natürlich das `\unskip` in `\beginpair` weglassen. Nun muß noch `\ILmarkup` angepaßt werden.

```
  \global\let\ILmarkupA = \ILmarkup
  \gdef\ILmarkup
      {\let   &\Midpair        % & startet als \Midpair
       \catcode`\&\active      % & aktivieren
      }
\endgroup
```

Some notations will be unfortunate even when
Manche Notationen werden unglücklich (gewählt) sein sogar falls
they are beautifully formatted.
sie sind wunderbar gesetzt.

Schlußbemerkungen (LaTeX)

Für den Gebrauch in der Praxis sollten diese Makros natürlich in die Datei mit den Definitionen bzw. in ein eigenes Style-File gesteckt werden, das unter LaTeX dann mittels `\usepackage` eingebunden werden kann. Wer darauf Wert legt, kann mittels

```
\newenvironment{interlinear}{\bIL}{\eIL}
```

noch für das typische LaTeX-Markup als Umgebung sorgen.

Sollen umfangreiche Texte im Interlinear-Stil gesetzt werden, wird man vielleicht auch überlegen, ob es sich auszahlt, die Effizienz der Makros auf Kosten

der Übersichtlichkeit geringfügig zu erhöhen, indem man in zweistufigen Makros die Aufrufe anderer Makros durch deren Expansion ersetzt beispielsweise `\bIL` und `\eIL` in der Definition der `interlinear` Umgebung, oder in `\interpairs`, das `\endpair` und `\beginpair` aufruft.

TEX für Serientäter

Peter Willadt

Dieser Artikel erläutert zuerst Anforderungen, die die Post an Serienbriefe stellt, anschließend werden Hinweise zur Optimierung der Verarbeitungsgeschwindigkeit seitens TEX und des Druckers gegeben.

Formalien

Für einzelne Briefe läßt die Post fast beliebige Freiheiten, was Aufmachung und Inhalt des Schriftstückes angeht. Will man jedoch in den Genuß von Portovergünstigungen kommen, so müssen die Regeln über maschinenlesbare Aufschriften und die Richtlinien für Infopost/Infobrief beachtet werden. Falls eine Sendung in Größe oder verwendetem Material vom Standard deutlich abweicht, ist eine Lektüre der Originalvorschriften kaum zu vermeiden. Im Zweifelsfall sollte der Post ein Muster zur Prüfung vorgelegt werden. Im folgenden wird davon ausgegangen, daß der Serienbrief auf hellem Papier gedruckt und in normale Fensterbriefumschläge (DIN lang) verpackt wird.

Die Anschrift

Für die Einhaltung der Postvorschriften winken Portoersparnisse in Höhe von mindestens 30 %, falls bestimmte Stückzahlen erreicht werden. Die Adresse des Empfängers muß in Buchstaben mit einer Versalhöhe zwischen 2,5 und 4,5 mm geschrieben sein. Für die gesamte Adresse darf nur eine Schriftart verwendet werden. Am liebsten ist der Post eine nicht proportionale Schrift, und zwar Pica oder Elite – also 10 oder 12 Zeichen pro Zoll. Falls Proportionalschrift verwendet wird, sind Unterschneidung und Ligaturen verboten, die Buchstaben

müssen einen Abstand von 0,3–0,5 mm haben. Ungewöhnliche Schriften[1] sind generell nicht erwünscht.

Der Leerraum zwischen zwei Wörtern soll 3–5 mm betragen. Alles ist mit normalem Zeilenabstand zu setzen, bis auf die Ortsangabe. Diese soll dem Rest der Anschrift mit einer Zeile Abstand folgen. Die Postleitzahl soll ohne Zwischenräume gesetzt werden, Postfachnummern hingegen sind in Zweiergruppen zu gliedern.

Die meisten Schriften aus der Computer-Modern-Familie sind also nicht besonders geeignet, da TEX seine Stärken eben gerade mit Unterschneidungen und Ligaturen zeigt. Die Schreibmaschinenschrift `cmtt10` hat den Nachteil einer für meine Begriffe zu großen Lauflänge – bei etwas komplizierteren Anschriften ist das Adreßfeld allzu schnell gefüllt. Wer sich die Mühe ersparen will, eine spezielle Parameterdatei für METAFONT und damit eine eigene Schrift zu erzeugen, kommt vielleicht mit `cmvtt10` ganz gut zurecht.

Der Inhalt

Neben Empfängernamen und Anschrift dürfen noch einige weitere Variablen in Serienbriefen verwendet werden. Allerdings gilt, daß alle Unterschiede, die sich durch verschieden große Wörter ergeben, zum Zeilenende auszugleichen sind.

Wird also ein variabler Text innerhalb eines mehrzeiligen Absatzes verwendet, dann muß der TEX-Umbruchalgorithmus daran gehindert werden, diesen Absatz optimal zu umbrechen, beispielsweise, indem eine Zeile mit dem Namen des Empfängers mit einem rüden `\\` beendet wird.

Optimierung

Hier geht es einerseits um die Optimierung der Verarbeitung durch TEX. Wenn eine Seite fast unverändert mehrere hundert Male gesetzt werden soll, lohnt es sich durchaus, über Optimierung nachzudenken. Andererseits geht es auch um Optimierung seitens des Druckertreibers und des Druckers. Hier bestehen Reserven vor allem dann, wenn Grafiken in Serienbriefe eingebunden werden.

Der Text

Der naivste Ansatz – den Text mehrmals einzulesen – hat seine Berechtigung, wenn die verwendete TEX-Version nur sehr wenig Speicherplatz zur Verfügung

[1] Zum Beispiel Fraktur oder aus dem cm-Fundus Schriften wie `cmdunh10`.

```
% Methode 1, Text einlesen.
\def\einbrief{
  \liesadresse
  \input brieftext
}
```

Figure 1: Serienbriefe auf die einfachste Art: Für jeden Brief wird der Text komplett vom Datenträger eingelesen.

stellen kann. Ansonsten gibt es effektivere Methoden, einen Text mehrmals auszugeben.

Möglichkeit zwei besteht darin, den gesamten Brieftext per `\def` im Speicher von TEX zu halten. Dagegen sprechen einige Gründe: Zum einen funktioniert das ganze nur, wenn im Text kein *outer* Befehl verwendet wird – sonst meldet TEX eine `Runaway definition`, zum anderen wird nicht nur der TEX-Speicher knapp, TEX muß zudem noch jedesmal den kompletten Text neu setzen. Falls nicht viele variable Textteile vorkommen, wird unnötig Rechenzeit verschwendet. Trotzdem ist dieses Verfahren deutlich schneller als das erste.

Die dritte Methode legt möglichst viele feste Textteile in *vboxes* und *hboxes* ab und kopiert sie jeweils nur für die einzelnen Briefe. TEX muß dadurch die einzelnen Absätze nicht mehr umbrechen, geschweige denn lesen. Innerhalb dieser im voraus gesetzten Absätze läßt sich dann allerdings auch kein variabler Text verwenden – aber dem stehen die postalischen Vorschriften ohnehin entgegen.

Dieses Verfahren erfordert nähere Beschäftigung mit dem Text und lohnt sich nur bei wirklich großen Stückzahlen, ansonsten wird der Computer zwar wenig belastet, der TEXniker verliert dafür aber um so mehr Zeit. Alle drei Methoden sind beispielhaft in den Abbildungen 1–3 skizziert.

Welche der Methoden letztlich gewählt wird, ist nicht zuletzt auch eine Frage des persönlichen Geschicks und Geschmackes. Zur Orientierung über den möglichen Zeitgewinn wurde ein Testdurchlauf nach allen drei Methoden gemacht. Die Ergebnisse finden Sie in Tabelle 1. Auf nähere Angaben über verwendeten Text und Systemumgebung wurde verzichtet, da die Ergebnisse auch so ganz gut vergleichbar sind.

Für TEXniker mit Vorlieben für exotische Lösungen gibt es noch eine vierte Methode, die hier kurz skizziert werden soll: In einem *virtual font* kann für jeden Buchstaben jeder beliebige Befehl, der auch in einer DVI-Datei stehen kann, verwendet werden. Damit ist es möglich, in einem *virtual font* einen

```
% Methode 2: Text in Makro
  \def\einbrief{
    \liesadresse
    Sehr geehrte\anrede,\par
    vielleicht haben...
    % viele Zeilen weiter
    mit freundlichen Gr"u"sen
    \unterschrift
    Fred Ferkel
    \neueseite
  }
```

Figure 2: Optimierung von Serienbriefen, erster Schritt: Anstatt den Text jedesmal einzulesen, wird er in ein Makro gepackt, das dann beim Setzen des Briefes aufgerufen wird.

Methode	*Seiten pro Sekunde*
(1) `\input`	67,5
(2) `\def`	126,5
(3) `\copy`	188,7

Table 1: Drei Methoden, Serienbriefe zu erzeugen, im Vergleich.

Buchstaben zu bauen, der aus einem ganzen Absatz besteht. TEX braucht dann nur einen Buchstaben anstelle eines Absatzes zu setzen und die DVI-Datei wird wesentlich kleiner. Allerdings wird die Last dadurch nur von TEX auf das DVI-Treiberprogramm verschoben. Da für einen Test dieser Methodik keine fertige Software zur Verfügung steht und ein Gewinn wohl nur dann erwartet werden kann, wenn entweder der DVI-Treiber auf einem wesentlich schnelleren Rechner als dem, auf dem das Dokument gesetzt wurde, läuft oder wenn nur ein Bruchteil der erzeugten Seiten benötigt wird, wurde hier auf die praktische Durchführung des Verfahrens verzichtet.

Grafik

DVI-Treiber sind gemeinhin bereits dahingehend optimiert, daß sie den Drucker mit maximaler Geschwindigkeit beliefern, sofern Drucker und Rechner von der Dimensionierung her zusammenpassen. Optimierungen sind erst dann erforderlich, wenn zum Text noch Grafiken hinzukommen. Kaum ein Serienbrief kommt ohne Grafik aus – sei es ein Logo oder auch nur eine gescannte Unterschrift.

```
% Methode 3: Aufsplitten
% Zuerst Boxen reservieren und f"ullen
\newbox\briefbox
\setbox\briefbox\vbox{
  vielleicht haben...
  % viele Zeilen weiter
  mit freundlichen Gr"u"sen
  \unterschrift
  Fred Ferkel
}
\newbox\anredenbox
\setbox\anredenbox\hbox{Sehr geehrte}
% und sie dann kopieren
\def\einbrief{
  \liesadresse
  \leavevmode\copy\anredenbox\anrede,\par
  \copy\briefbox
  \neueseite
}
```

Figure 3: Optimierung von Serienbriefen, zweiter Schritt: Absätze und andere Teile, die sich von Brief zu Brief nicht verändern, werden nur noch einmal gesetzt.

Sowohl DVI-Treiber als auch Drucker selbst sind auf eine schnelle Verarbeitung von Text ausgelegt. Hieraus ergibt sich ganz zwanglos, daß die schnellste Art, eine Grafik mehrmals auszugeben, darin besteht, sie in Schriftzeichen umzuwandeln. Hierfür gibt es einige Programme im TEX-Umfeld, beispielsweise `BM2FONT`. Auch TEX selbst kommt mit Lettern besser klar als mit Grafiken.

Das Umwandeln in Schrift funktioniert nur mit einfarbigen Grafiken. Grafiken mit Graustufen werden spätestens bei der Ausgabe auf Papier aufgerastert. Bei einem PostScript-Drucker ist es sinnvoll, das Aufrastern dem Drucker zu überlassen. Bei anderen Druckern kann überlegt werden, ob es sinnvoll ist, die Rasterung bereits vor der Einbindung in die DVI-Datei vorzunehmen, so daß das ‚Schriftzeichen' die gerasterte Grafik enthält. Falls es sich ursprünglich um eine Vektorgrafik handelte, nimmt man damit eine hohe Dateigröße und eine starke Geräteabhängigkeit in Kauf.[2] Ob dieses Vorgehen bessere oder

[2] Es wäre auch denkbar, mit einem Grafikprogramm eine Farbseparation vorzunehmen, die vier erhaltenen Graustufenbilder aufzurastern, als Schriftzeichen abzulegen und auf einem Farbdrucker übereinander zu drucken. Allerdings sind übliche Farbdrucker für den Hausgebrauch nicht auf diese Art von Anwendung ausgelegt. Es ist also mit Problemen zu rechnen,

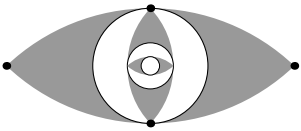


Figure 4: Ein Logo mit Graustufen

schlechtere Ergebnisse bringt als die Einbindung einer Grafik, die dann vom DVI-Treiber gerastert wird, hängt sowohl von der Grafik selbst als auch vom Treiber und der verwendeten Hardware ab. Pauschale Antworten lassen sich hier also nicht finden. Dem geneigten Leser bleibt nichts anderes, als selbst nachzumessen.

Bei PostScript-Druckern ergeben sich auch Möglichkeiten, Grafiken mit Graustufen oder mehrfarbige Grafiken soweit zu optimieren, daß nicht mit jeder Seite die immer gleiche Grafik erneut vom Computer her übertragen werden muß; unter Umständen kann sogar erreicht werden, daß der Drucker die Grafik nicht jedesmal neu zeichnen muß.

Auch hier ergeben sich mehrere Möglichkeiten, von denen ich zwei näher erläutern möchte. Beide setzen allerdings voraus, daß der PostScript-Code von Hand bearbeitet wird. Folglich gehört sorgfältig abgewogen, ob sich der Aufwand lohnt. Für das Logo der eigenen Firma oder die eigene Unterschrift ist diese Frage schnell beantwortet, bei einer einzelnen Mailing-Aktion sieht das ganze etwas anders aus. Bei komplizierten Grafiken lohnt sich der Aufwand generell eher nicht. Hier ist es sinnvoll, die Grafik separat zu drucken und hierfür die Kopierfunktion des Druckers zu benutzen oder – auch im Interesse der Druckqualität – bereits mit dem Logo bedrucktes Papier in einer Druckerei erstellen zu lassen.

Die Grafik selbst wird in Form mehrerer PostScript-Makros gespeichert. Falls der Drucker PostScript Level 2 versteht, werden die Makros in einer *Form*[3]

beispielsweise hinsichtlich der Vermeidung von zu hohem Farbauftrag und hinsichtlich der Farbtreue.

[3] Eine *Form* in PostScript ist eine Möglichkeit, eine Grafik mehrmals zu verwenden. Ein Drucker, der PostScript optimal nutzt, rastert die Form nur *einmal* auf und kopiert sie dann jeweils, anstatt sie jedesmal, wenn sie benötigt wird, erneut zu rastern. Eine Form verhält sich zu einem PostScript-Makro wie die oben bei der Optimierung des Textes geschilderte Methode 3 zur Methode 2.

abgelegt. Die Makros an sich sparen die mehrfache Übertragung vom Computer zum Drucker, die Form erspart dem Drucker das mehrfache Rastern der Grafik.

Die bearbeitete Grafik wird vor dem eigentlichen Dokument an den Drucker geschickt, bei DVIPS mit Kommadozeilen-Option `-h` oder durch Verwendung von `\special{`*header=datei.ps*`}`. An der Stelle, wo im TEX-Text die Grafik bisher als Datei eingebunden wurde, wird das Makro oder die Form entweder direkt per PostScript-Befehl aufgerufen oder es wird eine EPS-Datei eingebunden, die zusätzlich noch Angaben über die *Bounding Box* enthält. Der erste Weg ist deutlich schneller, aber etwas problematischer – beispielsweise muß man selbst den Platz für die Form freihalten, anstatt sich auf das Paket `graphics` berufen zu können.

Die eigentlichen Zeichenanweisungen werden per `def` in handliche Stücke zerlegt, die den PostScript-Interpreter nicht überfordern. Im Programmbeispiel wird per `languagelevel` unterschieden, ob PostScriptLevel 2 verwendet werden kann – falls ja, wird die Grafik als Form angesprochen, falls nicht, bleibt es bei `def`. Die innerhalb des Briefes einzubindende Grafikdatei wird dadurch wesentlich verkürzt und sieht etwa so aus – als Beispiel verwende ich Abbildung 28 aus dem Handbuch zu METAPOST und unterstelle, daß DVIPS und `graphics.sty` verwendet werden:

```
%!PS
%%BoundingBox: -57 -32 57 32
% File logobody.ps
zeichnelogo
%%EOF
```

Innerhalb des Briefes wird dann an der Stelle, an der das Logo erscheinen soll, `\includegraphics{logobody.ps}` geschrieben. Statt dessen kann natürlich im Brief selbst auch per `\special{"zeichnelogo}` die Funktion direkt aufgerufen werden, was dann nochmals etwas TEX-Laufzeit spart. Die Datei, die im Vorspann eingebunden wird, enthält die ganzen Zeichenanweisungen – siehe Abbildung 5.

Der Benutzer des Logos muß nicht darüber nachdenken, welcher PostScript-Sprachumfang von seinem Ausgabegerät verwendet wird, es wird automatisch umgeschaltet. Die Ergebnisse der PostScript-Optimierung finden Sie in Tabelle 2. Anstatt die Messungen mit einem PostScript-Drucker durchzuführen, habe ich, schon der Papierersparnis wegen, Ghostscript rastern lassen und die Ausgabe in eine Datei geschickt. Die erhaltenen Ergebnisse spiegeln dadurch leider nicht ganz den möglichen Zeitgewinn wider.

```
%!PS
% File logohead.ps
/logostroke {
0.6 setgray newpath 54 0 moveto
39.44084 13.86606 20.10564 21.60022 0 21.60022 curveto
-20.10564 21.60022 -39.44084 13.86606 -54 0 curveto
% viele weitere Zeilen aus dem MP-Manual Abb. 28
newpath 0 -21.60022 moveto 0 0 rlineto stroke
% aber showpage und %%EOF auslassen
} def
languagelevel 2 ge {
  % Level 2
  /Logoform 6 dict def
  Logoform begin
   /Formtype 1 def
   /BBox  [-57 -32 57 32] def
   /Matrix [1 0 0 1 0 0] def
   /PaintProc {
    begin gsave logostroke
    grestore end
   }def
  end
  /zeichnelogo {Logoform execform} def
}{
  % Level 1
  /zeichnelogo { gsave logostroke grestore} def
}ifelse
%%EOF
```

Figure 5: Optimierung von EPS-Grafiken. Die Zeichenanweisungen werden in einer Datei abgelegt, die lediglich einmal pro DVI-Datei eingebunden wird.

Methode	*Seiten pro Sekunde*		
	TEX	*dvips*	*Ghostscript*
(A) Logo einlesen	29	26	4
(B) Logo im Vorspann	87	50	4

Table 2: Zwei Methoden zur Grafikeinbindung im Vergleich. Neben der TEX-Zeit sind hier auch die Laufzeiten für `dvips` und Ghostscript, letzteres stellvertretend für einen PostScript-Drucker, angeführt.

Die Druckgeschwindigkeit wird oft durch den Durchsatz der Schnittstelle vom Computer zum Drucker begrenzt. Die nach Methode B (Definition im Vorspann) erzeugte Datei ist typischerweise wesentlich kleiner als die der Methode A. Da Ghostscript auf dem Computer läuft, geht dieser Faktor in die Meßergebnisse nicht ein. Zudem scheint Ghostscript, zumindest in der von mir verwendeten Version 5.50, keinen Gebrauch von der möglichen Optimierung durch Verwendung von *form* zu machen.

EPS-Optimierung näher betrachtet

Analog zu TEX ist PostScript eine ausgewachsene Programmiersprache. Dies ist hinsichtlich der Grafik-Optimierung Segen und Fluch zugleich. Viele Grafikprogramme erzeugen EPS-Dateien, die neben den Zeichenanweisungen ein mehr oder minder umfangreiches Makropaket enthalten. Oftmals ist das Makropaket wesentlich größer als die Grafik selbst. Die Identifizierung der eigentlichen Anweisungen zum Zeichnen wird dadurch nicht einfacher. Gescannte und anschließend vektorisierte Bilder kommen meist mit einer Handvoll PostScript-Anweisungen aus. Mit guten PostScript-Kenntnissen und viel Geduld ist es möglich, ein Makropaket auf diejenigen Funktionen zurechtzukürzen, die tatsächlich für das Zeichnen der Grafik benötigt werden. Werden die EPS-Dateien immer vom selben Programm erzeugt, muß die sorgfältige Analyse nur einmal durchgeführt werden. Bei weiteren Grafiken reicht es dann aus, das automatisch erzeugte Makropaket per Texteditor durch das von Hand gekürzte zu ersetzen. Ordentlich erstellte EPS-Dateien enthalten sogenannte DSC-Kommentare[4], die mit zwei `%`-Zeichen am Zeilenanfang eingeleitet werden. Sie erleichtern die Erkennung, welche Teile der EPS-Datei welchen Ursprung haben und unterstützen so die Optimierung. Da METAPOST kein Makropaket erzeugt, erspart es viele Mühen bei der Nachbearbeitung.

[4] DSC=document structuring conventions

Es ist oft nicht möglich, sämtliche Anweisungen zum Zeichnen einer Grafik in einem einzigen PostScript-Makro abzulegen, da die Makrogröße begrenzt ist. Zu allem Übel ist die maximale Größe geräteabhängig. Daher empfiehlt es sich, einzelne überschaubare Makros zu definieren. Diese können dann wiederum von einem anderen Makro aufgerufen werden. Die Umschaltung zwischen Form und Makro kann, wie in Abbildung 5 gezeigt, direkt übernommen werden. In den Zeilen nach `languagelevel 2 ge {` müssen lediglich in dem die *Bounding Box* betreffenden Befehl `/BBox` die Abmessungen der Grafik korrigiert werden.

Schlußbemerkung

Für den Hausgebrauch ist TEX auf heute üblichen Rechnern allemal ausreichend schnell. Wo große Mengen von Serienbriefen erzeugt werden sollen, besteht aber durchaus noch Raum für Optimierungen, wobei die Zeitersparnisse gegenüber dem erhöhten Aufwand bei der Erstellung der Eingabedateien abgewogen werden müssen. Falls genug TEX-Speicher zur Verfügung steht, empfiehlt sich die Umwandlung der Einbindung einer Datei per `\input` in eine Makrodefinition per `\def`. Die Ergebnisse dieser einfach durchzuführenden Optimierung können noch weiter verbessert werden. Der mit weiteren Verbesserungen verbundene Aufwand ist allerdings unverhältnismäßig hoch.

Werden Logos oder andere Grafiken eingebunden, so kann alleine eine Optimierung dieser Einbindung eine Vervielfachung sowohl der Geschwindigkeit, mit der TEX arbeitet, als auch der, mit der die DVI-Datei weiterverarbeitet wird, nach sich ziehen. Wird ein Logo öfters verwendet, so ist dessen Optimierung also dringend zu empfehlen.

4allTEX-Preview – TEX für alle?

Gerhard Wilhelms

4allTEX 5.0 ist eine TEX-Umgebung für Windows. Diesem Bericht liegt eine Vorab-Version zugrunde. Kleine Unzulänglichkeiten dieser Version wurden umgehend vom Autorenteam Erik Frambach/Wietse Dol abgestellt. Der Bericht soll als Entscheidungshilfe dienen, ob 4allTEX als Alternative für das eigene System in Frage kommen kann. Ein erweiterter Praxistest der endgültigen Version von 4allTEX folgt in der nächsten Ausgabe von „Die TEXnische Komödie“.

Einleitung

Als TEX-Umgebung für Windows muß sich 4allTEX einem harten Windows-Anwender-Test stellen: Installation des Systems und erste Arbeiten, ohne eine einzige Zeile der Anleitung zu lesen. Dieses Testprofil erschien mir auf Grund der Erfahrungen mit der Plug&Play-TEX-Version für Windows des Rechenzentrums der Universität Augsburg angemessen. Hier zeigte die Erfahrung, daß nur ca. 5 % der Anwender überhaupt einen Blick in die Anleitung werfen. Mit der gleichen Blauäugigkeit bin ich an 4allTEX herangegangen und protokolliere hier meine Erlebnisse. Als Testsystem diente ein Pentium II 350 mit Windows 98.

Installation

Der erste Test wurde mit Bravour bestanden: CD-ROM in das Laufwerk eingelegt und das Installationsprogramm startet automatisch. Man wird mit dem Installationsfenster aus Abbildung 1 begrüßt.

Die Sprache läßt sich leider (noch?) nicht auf „Deutsch“ umstellen. Die Voreinstellung der einfachen Installation (simple) verlockt, so daß ich mit der Schaltfläche „Next“ fortgefahren bin. Das nächste Fenster enthält die erste Überraschung, eine Uninstall-Anweisung. Spätere Recherche hat ergeben, daß es sonst keine Deinstallationsroutinen gibt, weder im Startmenü noch unter *Einstellungen → Software*. Erfreulich dagegen die Möglichkeit, das System direkt von der CD-ROM starten zu können oder zur Performanzsteigerung auf die Festplatte kopieren zu können.

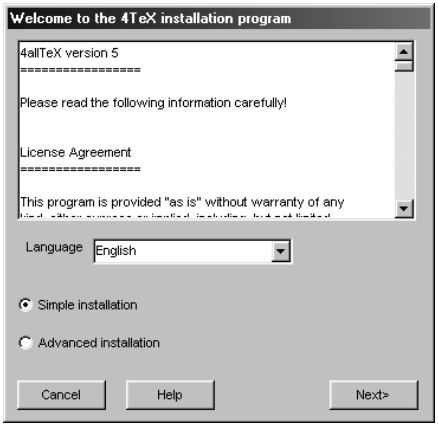


Figure 1: Installation

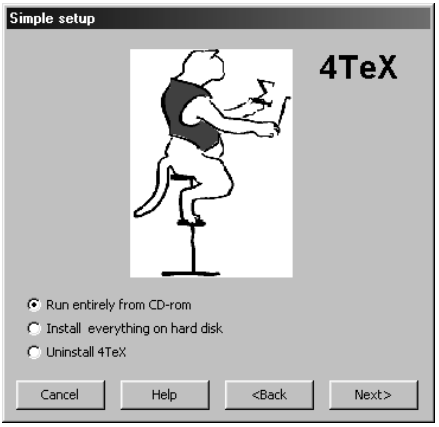


Figure 2: Installationsoptionen

Da ich das System zunächst testen wollte, erschien mir die CD-ROM-basierte Installation angebracht. Nach kurzer Wartezeit für das Kopieren einiger Systemdateien und Anlegen von Ordnern für die dynamisch erzeugten Zeichensätze wurde ich vom Hauptfenster von 4allTEX begrüßt, das Sie in Abbildung 3 sehen.

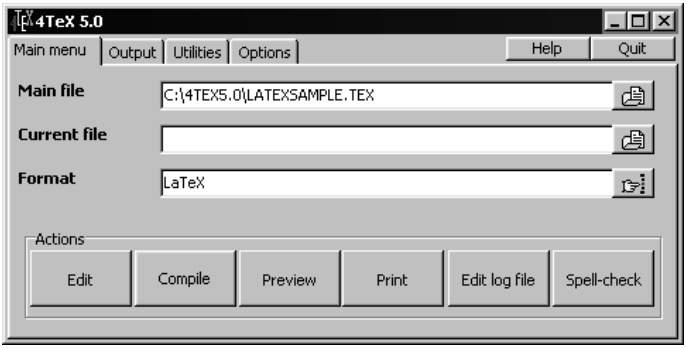


Figure 3: Die Benutzeroberfläche von 4allTEX

Mein erster Blick galt den Optionen und erfreut stellte ich fest, daß hier auf die deutsche Sprache umgestellt werden konnte, wie Sie in Abbildung 4 sehen.

Figure 4: Die Einstellungsmöglichkeiten von 4allTEX

Zudem lassen sich drei verschiedene Editoren (PFE, WinEDT und MED; letzterer inklusive Nutzungslizenz, die anderen Shareware) in das System einbinden, aus denen ich meinen persönlichen Favoriten WinEdt auswählte. Leider läßt das 4allTEX-System nur aus einer Liste auswählen, so daß ich zwangsweise auf die englische Version von WinEdt auf der CD-ROM festgenagelt wurde und nicht meine eigene, angepaßte Version auf der Festplatte in die TEX-IDE einklinken konnte.

Der Blick in das Utility-, Pardon Hilfsprogramme-Menü, läßt erahnen, warum 4allTEX auf zwei CD-ROMs ausgeliefert wird. Nichts, was der fortgeschrittene Unix-TEX-Nutzer kennt und liebt, fehlt auf dieser Windows-CD-ROM, wie Abbildung 5 belegt.

Die Liste ist viel zu umfangreich, um alle Programme aufzuzählen, doch neben Perl und Emacs, den Gnu-Tools, PostScript-Utilities, Ghostscript, diversen Zeichenprogrammen und Grafikbearbeitungstools finden sich auch der Acrobat-Reader und Paint Shop pro.

Fazit: Installation und Lieferumfang machen einen ausgesprochen guten Eindruck!

Figure 5: Die in 4allTEX integrierten Programme

Benutzung – zum ersten

Als erster Härtetest sollte ein mit der Koma-Script-Briefklasse geschriebenes Dokument überprüfen, ob auch „ungewöhnliche“ Klassen auf der CD-ROM sind und die deutschen Trennmuster korrekt eingebunden sind. Die Übersetzung mit LaTeX wird, ähnlich wie bei MiKTeX, in einem gesondert geöffneten Fenster protokolliert. Allerdings konnte ich nur kurz erkennen, daß eine HugeTeX-Version zum Einsatz kam, denn das Fenster wurde umgehend wieder geschlossen. Keine `.log`-Datei und keine `.dvi`-Datei! Meine erste Vermutung war, daß die CD-ROM-basierte Installation wohl fälschlicherweise Dateien auf der CD-ROM anlegen möchte und wegen des Schreibschutzes nicht kann. Nächster Akt: Installation auf Festplatte.

Die Profi-Installation (advanced)

Die Installation im Profi-Modus macht wieder einen ausgezeichneten Eindruck. Die einzelnen Programmpakete werden als sogenannte Module in drei Kategorien eingeteilt – „required“, „recommended“ und „optional“, also notwendig, empfohlen und zusätzlich möglich. Die CD-ROM „TeX Live“ läßt grüßen! Besonders gefällt bei der Auswahl, daß das Aktivieren eines Moduls, das auf anderen Modulen basiert bzw. nur mit anderen Modulen in Zusammenarbeit funktioniert, diese anderen Module automatisch mit aktiviert werden. Außerdem wird der benötigte Festplattenplatz in Abhängigkeit der Cluster-Größe berechnet. Windows-95-Nutzer mit ungeschickt eingerichteten Partitionen (Cluster-Größe 32 KByte) können so über TeX-Installationen mit weit über 600 MB Speicherbedarf staunen, während Windows-98-Nutzer mit 8 KByte-Clustern mit gut 200 MB vergleichsweise günstig davonkommen. Der *slack* schlägt unbarmherzig zu, da sich TeX aus einer Vielzahl kleiner und kleinster Dateien zusammensetzt.

Der Installationsvorgang zieht sich ziemlich in die Länge, unterhält aber mit einen Fortschrittsbalken mit immer verschiedenen Farbverläufen. Ich hätte mir allerdings gewünscht, daß der korrekt ermittelte Speicherbedarf zu einer korrekt umgesetzten Fortschrittsanzeige führt, nicht einem immer wieder neu aufgesetzten Balken.

Benutzung – zum zweiten

Auch die festplattenbasierte Installation führte in der Anwendung nicht zum gewünschten Erfolg. Die Übersetzung der TeX-Quelle startete, das Protokollfenster wurde unvermittelt geschlossen, ohne daß nur die geringste Chance

bestand, den Grund des Abbruchs abzulesen, und ich saß hilflos ohne `.log`-geschweige denn `.dvi`-Datei da. Spätestens jetzt würde der typische Windows-Anwender frustriert die Deinstallation des Systems starten.

Fazit

Die wirklich professionelle Installationsroutine und der außergewöhnlich umfangreiche Lieferumfang wissen bei 4allTEX zu gefallen. Daß dann die Shell TEX so startet, daß es die Format-Datei nicht findet, und man nur durch Starten von TEX auf der Kommandozeile überhaupt darauf kommt, was der Grund für dieses Verhalten ist, machen das System leider für Anfänger ungeeignet.

Derlei kleine Fehler sind sicher leicht zu beheben und auch der „Konkurrenz“ TEX Live nicht unbekannt. Hier hilft nur eine verlängerte Testphase[1]. Manchmal ist ein Produkt mit den aktuellsten Daten eben doch nicht einem ausgereiften und getesteten Produkt vorzuziehen, insbesondere wenn man neue Nutzer gewinnen will.

Trotz des kleinen Ausrutschers gehört 4allTEX meiner Meinung nach zu den besten TEX-Distributionen für Windows. Aber eben nicht für Anfänger ...

[1] Anmerkung der Redaktion: In diesem Beitrag wurde eine Vorversion getestet. Die endgültige Version scheint die beschriebenen Probleme nicht mehr zu haben.

TEX-Beiprogramm

Erratum für die CTAN-CD-ROM von DANTE e.V.

Klaus Höppner

Wichtiger Hinweis für Nutzer von Windows 9x/NT

Wegen eines Fehlers in der Version des Programmes `mkisofs`, die zum Erstellen der CD-ROM vom Mai 1999 benutzt wurde, zeigt Windows Explorer in jedem Verzeichnis der CD-ROM eine Datei `<translation table>` an, die sich nicht öffnen läßt. Leider lassen sich dadurch keine Verzeichnisbäume über *Drag & Drop* im Windows Explorer von der CD-ROM auf die Festplatte kopieren, da der Kopierprozeß mit der Fehlermeldung abbricht, daß die Dateien `<translation table>` nicht kopiert werden können.

Das Kopieren kompletter Verzeichnisbäume kann folgendermaßen durchgeführt werden:

1. Öffnen eines MS-DOS-Fensters (*Start* → *Programme* → *MS-DOS Eingabeaufforderung*)
2. Kopieren des Verzeichnisbaums mit

   `xcopy /s /c` *`Quellverzeichnis Zielverzeichnis`*

   Beispiel: Besitzt das CD-ROM-Laufwerk den Laufwerkbuchstaben `D:` und Sie wollen das Verzeichnis `macros\latex\required` mit allen Unterverzeichnissen von der CD-ROM in das Verzeichnis `foo` auf der Festplatte (`C:`) kopieren, so tippen Sie

   ```
   xcopy /s /c d:\macros\latex\required c:\foo
   ```
3. Falls das Zielverzeichnis noch nicht existiert, fragt Windows, ob es sich bei dem Ziel um ein Verzeichnis oder eine Datei handelt. Geben Sie hier `V` für Verzeichnis ein. (Dieser Buchstabe gilt nur für die deutsche Version von Windows!)

4. Beim Kopieren erscheinen Warnungen, daß die Datei `<translation table>` nicht kopiert werden konnte. Diese können ignoriert werden.
5. Eine Beschreibung der Optionen von `xcopy` erhalten Sie durch Eingabe von

   ```
   xcopy /? | more
   ```

Zitat

> Nicht nur Schriftarten, Schriftschnitte, Schriftgrade oder die Satzausrichtung sind typographische Stilelemente, sondern ebenso die Strichstärke der Linien, die zwischen Textspalten, als Abgrenzung zwischen Abschnitten oder in Tabellen benutzt werden. Ja selbst die Art, wie Strichzeichnungen angelegt, wie Abbildungen und Tabellen im Gestaltungsraster plaziert werden, sind Stilelemente. Gehen Sie mit diesen so sparsam um, wie mit den Schriftstilen. Sparsam muß nicht geizig sein.
>
> *Jürgen Gulbrins,*
> *Christine Kahrmann:*
> *Mut zur Typographie.*
> *Berlin, Heidelberg 1992.*

Spielplan

Termine

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

19.9.1999 21. Mitgliederversammlung von DANTE e.V.
Universität Heidelberg
Kontakt: DANTE e.V.

20.9.–24.9.1999 EuroTEX'99 – 11th European TEX Conference
Universität Heidelberg
Kontakt: EuroTEX'99

23.9.–24.9.1999 H2PTM99 – 5th Conference on Hypertexts and Hypermedia: Products, Tools, Methods
Saint Denis, Paris, Frankreich
Kontakt: Prof. Imad Saleh

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

6.12.–9.12.1999 XML'99
Pennsylvania Convention Center, Philadelphia, PA, USA

8.3.–10.3.2000 DANTE 2000
Universität Clausthal-Zellerfeld
Kontakt: Jan Braun

12.8.–18.8.2000 TUG'2000 – "TEX enters a new millenium"
21st annual meeting of the TEX User Group
Wadham College, Oxford, UK
Kontakt: TEX Users Group

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

Erlangen – Walter Schmidt
`walter.schmidt@arcormail.de`
Peter Seitz
`p.seitz@koehler-seitz.de`
Online-Café, Hauptstrasse 55–57 (Altstadtmarkt)
Dritter Dienstag im Monat, 20.00 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

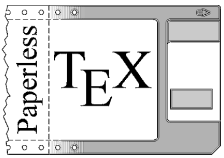
Karlsruhe – Klaus Braune
Tel.: 07 21/6 08 40 31
`braune@rz.uni-karlsruhe.de`
Universität Karlsruhe, Rechenzentrum
Zirkel 2, 3. OG Raum 316
Erster Donnerstag im Monat, 19.30 Uhr

Stuttgart – Marcus Schweizer
Tel.: 07 11/6 85 44 44
`schweiz@theochem.uni-stuttgart.de`
Gaststätte „Alte Mira“, Büchsenstr. 24
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

EuroTEX'99 – XI. Europäische TEX-Konferenz



Einladung zur Teilnahme

Die XI. Europäische TEX-Konferenz wird vom 20. September bis zum 24. September 1999 in Heidelberg stattfinden. Das Thema der Konferenz lautet *Papierloses TEX* (Paperless TEX). Hierzu hat das Programm-Komitee eine Reihe von spannenden Vorträgen zusammengestellt, die die neuesten Entwicklungen und Erkenntnisse aus dem Bereich Online-Publishing mit TEX und seinen Komponenten im wissenschaftlichen und technischen Umfeld präsentieren. Die Konferenzsprache ist hauptsächlich englisch, einzelne Vorträge oder Meetings finden in anderen europäischen Sprachen statt. EuroTEX'99 wird ausgerichtet von der Universität Heidelberg, vertreten durch das Psychologische Institut und das Universitätsrechenzentrum. Tagungsort ist das Psychologische Institut im Herzen der Altstadt von Heidelberg.

Alle Mitglieder von DANTE e.V. sind herzlich zur Teilnahme eingeladen, aktuelle Informationen finden Sie unter `http://EuroTeX99.uni-hd.de`.

Programmübersicht

Sonntag, 19.9.	14:00–18:00	Anmeldung
	14:00–18:00	Mitgliederversammlung DANTE e.V.
Montag, 20.9.	08:00–09:30	Anmeldung
	09:30–18:00	Begrüßung, Vorträge, Workshops, etc.
	ab 18:00	BoF, spezielle Meetings, etc.
Dienstag, 21.9.	09:30–15:00	Vorträge, Workshops, etc.
	15:30–18:30	Schiffahrt nach Hirschhorn mit Buffet
	18:30–20:00	Stadtbesichtigung, Rückfahrt mit Zug
Mittwoch, 22.9.	09:30–16:00	Vorträge, Workshops, etc.
	ab 18:00	Empfang beim Rektor der Universität mit anschließendem Abendessen
Donnerstag, 23.9.	09:30-18:00	Vorträge, Workshops, etc.
	ab 18:00	BoF, spezielle Meetings, etc.
Freitag, 24.9.	09:30–18:00	Tutorien

Vorläufige Liste der Referenten/Themen

- Jacques André, Hélène Richy: Paperless preparing or editing manuscripts and proofreading
- Sasha Berdnikov: Old Slavonic and Church Slavonic in TEX and Unicode
- Sasha Berdnikov: METAFONT to Type1 font conversion, parallel design of METAFONT/Type1 fonts and related topics
- Erdmuthe Meyer zu Bexten, Jens Hiltner: LATEX: Das ideale Textverarbeitungssystem für blinde Studierende in naturwissenschaftlichen Disziplinen?!
- Włodek Bzyl: Detection and correction of spelling errors in marked up documents
- Hans Hagen: The NTG MAPS bibliography, a tour on converting SGML into advanced PDF using TEX
- Hans Hagen: What way are we heading, some thoughts on authors, readers and design
- Taco Hoekwater: An Extended Math Font Set For Processing MathML
- Bogusław Jackowski, Janusz Nowacki: Antykwa Półtawskiego – A parameterized outline font
- Jens Klöcker, Jörg Knappen: Is LATEX 2_ε markup sufficient for scientific articles?
- Jens Klöcker, Jörg Knappen: Creating a new standard document class for scientific articles
- Olga Lapko, Irina Makhovaia: "CM" and "TLC" in Russian – Production notes
- Anselm Lingnau: TkDVI: DVI Previewing in Tcl/Tk Programs – Current Status
- Dick Nickalls: MathsPIC – a PICTEX pre-processor for drawing geometry figures
- The $\mathcal{NTS}$ team: $\mathcal{NTS}$ – From conception to implementation
- Heiko Oberdiek: PDF information and navigation elements with hyperref, pdfTEX and thumbpdf
- Karel Píška: Fonts for Neo-Assyrian cuneiform

- Kristoffer Høgsbro Rose: Towards a DTD for LaTeX
- Sergey Strelkov: The Testbed for Preparation of Russian Patent Document in the XML Format
- Valentin Zaitsev, Andrew Janishewsky: Russian typographical traditions in mathematical literature

Tutorien

- Ulrik Vieth: Surviving the TEX font encoding mess, or: Understanding the world of fonts and mastering *fontinst*
- Klaus Braune: Graphics for TEX and pdfTEX
- Bogusław Jackowski: METAPOST – A universal graphic tool
- N.N.: Working with WinEdt

Teilnahmegebühren

Mitglieder der TUG oder einer anderen lokalen TEX-Organisation	190 €
Studenten (kein Tagungsband) mit entsprechendem Nachweis	90 €
Nicht-Mitglieder	240 €

Im Preis enthalten sind

- die „Social Events“ (am Dienstag und Mittwoch mit Abendessen),
- Getränke in den Pausen,
- Tagungsband,
- Tutorien (Freitag).

Zur Anmeldung füllen Sie bitte das Anmeldeformular unter `http://EuroTeX99.uni-hd.de` aus und überweisen den Tagungsbeitrag auf das dort angegebene Konto. Falls Sie nicht über einen WWW-Zugang verfügen, senden wir Ihnen die Unterlagen gerne per E-Mail oder Post zu. Bitte wenden Sie sich mit allen Fragen an:

EuroTEX'99
c/o Psychologisches Institut
Hauptstr. 47–51
69117 Heidelberg
Tel.: 06221/547267
Fax: 06221/547734
E-Mail: `eurotex99@urz.uni-heidelberg.de`

TEX-Tagung DANTE 2000 in Clausthal-Zellerfeld — Ankündigung und Call for Papers

Jan Braun

Die TEX-Tagung DANTE 2000 findet

vom 8. bis 11. März 2000 an der Technischen Universität Clausthal

statt. Die Veranstalter sind gemeinsam das Institut für Technische Mechanik und das Rechenzentrum der Technischen Universität Clausthal sowie DANTE, Deutschsprachige Anwendervereinigung TEX e.V.

Am Mittwoch sind Tutorien geplant, Donnerstag und Freitag sind für Vorträge, Diskussionen und Präsentationen vorgesehen, am Samstag wird voraussichtlich die Mitgliederversammlung von DANTE e.V. stattfinden.

- Alle, die einen *Vortrag* oder ein *Tutorium* halten oder eine *Diskussion* leiten möchten, werden gebeten, dies mit dem Formular „Anmeldung von Beiträgen“ (`http://dante2000.itm.tu-clausthal.de/CfP/`) oder per E-Mail (`dante2000@itm.tu-clausthal.de`) an den Organisator bis zum

  1. November 1999

  anzumelden. Die Annahme von verspäteten Anmeldungen ist nur unter Vorbehalt möglich.

  Zu einem Vortrag oder Tutorium ist ein *extended abstract* einzureichen. Richtlinien dafür sind auf der oben genannten WWW-Seite zu finden oder beim Organisator zu erfragen.

- Alle Firmen und Institutionen, die ihre Produkte präsentieren bzw. die Tagung finanziell unterstützen wollen, werden gebeten, sich möglichst frühzeitig an dieselben Adressen zu wenden.

- Die *Homepage* der Tagung findet sich unter

  `http://dante2000.itm.tu-clausthal.de/`

  oder

  `http://www.dante.de/dante2000/`

- Mit *Fragen*, Wünschen und Anregungen oder auch wohlbegründeter Kritik wenden Sie sich bitte an

Jan Braun
DANTE 2000
Institut für Technische Mechanik
Technische Universität Clausthal
Graupenstraße 3
D-38678 Clausthal-Zellerfeld
Tel: +49 5323 72-2073
Fax: +49 5323 72-2203
E-Mail: `dante2000@itm.tu-clausthal.de`

oder an DANTE e.V. in Heidelberg.

Die Veranstalter hoffen, daß möglichst viele TEX-Interessierte unsere Veranstaltung in Clausthal-Zellerfeld besuchen werden, und freuen sich auf einen erfolgreichen Tagungsverlauf.

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	
	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` [134.93.8.251]
E-Mail:	`ftpmail@dante.de`
WWW:	`http://www.dante.de/`
Mailbox:	0 62 21/16 84 26

Die TEXnische Komödie

11. Jahrgang Heft 3/1999 August 1999