DANTE

Deutschsprachige Anwendervereinigung TEX e.V.

# Die TEXnische Komödie

Heft 4/1995 7. Jahrgang März 1996



## Impressum



Beiträge sollten in Standard-LATEX-Quellcode 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: `emTeX (tex386), Version 3.14159 [4b]`, `LaTeX2e <1995/12/01> patch level 1`, `ghostview` (für die Bildschirmdarstellung) und `dvipsk 5.58f` (für Korrektur und endgültige Belichtung).

Erscheinungsweise: vierteljährlich
Erscheinungsort: Heidelberg
Auflage: 3000

Herausgeber: DANTE, Deutschsprachige Anwendervereinigung TEX e.V.
Postfach 10 18 40
69008 Heidelberg
Tel.: 0 62 21/2 97 66
Fax: 0 62 21/16 79 06
e-mail: `dante@dante.de`

Druck: PrintArt GmbH
Kirchenstr. 8
67125 Dannstadt



Redaktionsschluß für Heft 1/1996: 30.4.1996

# Editorial

Liebe Leserinnen und Leser,

mit dieser Ausgabe wird eine Änderung manifestiert, die bisher nur den Aufmerksamen unter Ihnen aufgefallen sein dürfte: Es gibt eine eigene LaTeX2-Klasse für „Die TeXnische Komödie“. Es mögen zwar noch kleinere Änderungen hinzu kommen – bei welchem Makro wäre das nicht so? – aber im Großen und Ganzen hat die neue Klasse ihre Kinderkrankheiten überwunden und ist jetzt einsatzfähig. Die Hauptarbeit dabei wurde von Gerd Neugebauer geleistet, bei dem ich mich nochmal herzlich dafür bedanken möchte.

Auch eine weitere Neuerung macht Fortschritte: das veränderte Aussehen der Mitgliederzeitung. Mit einer der nächsten Ausgaben beginnen wir, das neue, lang angekündigte Layout zu verwenden. Wir hoffen, Sie damit hoffentlich angenehm überraschen zu können. Ich bin schon jetzt auf die Reaktionen dazu gespannt.

In diesem Heft finden Sie einen weiteren Teil von Bernd Raichles Serie zu den Eigenwilligkeiten von TeX und einen Artikel von Werner Lemberg zum Einsatz von nicht-lateinischen Schriften in TeX. Beide Artikel zeigen, wie vielseitig es einsetzbar ist. Aber auch der Humor kommt nicht zu kurz, wie man im Beiprogramm sehen kann. Versäumen Sie außerdem nicht, sich die Ankündigung zur Tagung TUG'96 anzusehen, die in diesem Jahr in Rußland stattfinden wird.

Ich hoffe, daß auch diesmal wieder für Alle etwas dabei ist und verabschiede mich bis zum nächsten Mal

Ihre Luzia Dietsche





# Hinter der Bühne

**Vereinsinternes**

## Grußwort

Liebe Mitglieder,

wie immer an dieser Stelle möchte ich die Ereignisse seit dem Erscheinen der letzten Ausgabe der Mitgliederzeitung zusammenfassen.

All diejenigen, die große Neuigkeiten erwarten, muß ich dieses Mal enttäuschen. Es wurde viel getan, aber es war nichts darunter, das besonders erwähnenswert wäre. Dies ist nicht überraschend, da seit dem Erscheinen der letzten TeXnischen Komödie nicht viel Zeit vergangen ist.

Die größte Veränderung gab es bei den Angestellten des Vereins. Wir haben eine neue Arbeitskraft eingestellt, die sich bevorzugt um unsere Finanzen kümmern wird. Dies war notwendig geworden, weil es sich zeigte, daß der Arbeitsaufwand bei der Abrechnung mittlerweile so groß war, daß sie nicht mehr, wie in der Vergangenheit, ehrenamtlich erledigt werden konnte. Wir gehen davon aus, daß diese Entlastung, die vor allem unserem Schatzmeister zu Gute kommt, hilft, unseren Rückstand beim Kassenbericht, der auch beim Finanzamt für die Gemeinnützigkeit benötigt wird, relativ schnell aufzuholen.

Wie geplant, bekamen wir 500 Exemplare der CD-ROM von dem Verlag Addison-Wesley und konnten sie wunschgemäß an unsere Mitglieder zum Preis von 10,–DM weitergeben. Der einzige Wermutstropfen in diesem Fall war, daß Addison-Wesley von dem Erhalt der „Master“-Kopie bis zur Auslieferung unserer Exemplare mehr als zwei Monate benötigte. Es besteht jedoch Hoffnung, daß sich dies in der Zukunft verbessert.

Andererseits erhielt ich in dieser Woche ein Exemplar der InfoMagic CD-ROM, die in Deutschland vom Franzis-Verlag in den Buchhandel gebracht wurde. Sie enthält ebenfalls eine Kopie des CTAN-Archivs. Allerdings wurde auch hier, wie bei der CD-ROM von DANTE e.V., auf einen Teil der allgemein verfügbaren Software verzichtet. Da InfoMagic die Software nicht komprimiert hatte, wurden zwei CD-ROM benötigt. Weniger erfreulich ist die Tatsache, daß die Verpackung mit dem Vermerk „Copyright 1996“ vorgaukelt, es handle sich um

einen neuen Abzug, obwohl die CD-ROM den Stand vom September 1995 hat. In meinen Augen ist solch eine nur auf Kommerz ausgerichtete Vorgehensweise ein schlechter Service am Kunden. Ärgerlicherweise schadet diese Vorgehensweise dem Ruf und der Akzeptanz von TEX.

Den erfreulichen Gegensatz dazu stellt die Firma Suse dar, die uns stets, wenn sie eine neue Version der LINUX-CD-ROM herausbringt, ein Belegexemplar zusendet. Hierbei handelt es sich zwar nicht um eine spezielle TEX-CD-ROM, es ist jedoch ein TEX-System auf ihr enthalten. Diese Firma bemüht sich sehr, ihren Kunden einen möglichst neuen Stand zu bieten.

Zur Zeit laufen die Vorbereitungen für DANTE'96 in Augsburg auf Hochtouren. Die Hauptarbeit leisten wie immer der lokale Vertreter der Universität Augsburg, Herr Dr. Wilhelms, und auf der Seite von DANTE e.V. unsere Schriftführerin, Frau Dietsche. Das Programm enthält viele interessante Vorträge zu neuen Entwicklungen in TEX und METAFONT, Wissenswertes über Typographie, PostScript und vieles aus dem Umfeld zu TEX. Das Rahmenprogramm hat auch einiges zu bieten.

Auf der Mitgliederversammlung nächste Woche hoffe ich, wie auch schon in der Vergangenheit, viele von Ihnen anzutreffen.

Joachim Lammarsch
(Präsident)





**Bretter, die die Welt bedeuten**

## Eine Klasse für „Die TEXnische Komödie“

Gerd Neugebauer

„Die TEXnische Komödie“ wird neuerdings mit LATEX 2ε hergestellt. In diesem Artikel werden einige Interna der benutzten Dokumentenklasse erläutert: Wie sie entstand, welche Möglichkeiten sie bietet und wie man sie benutzen kann.

### Grundsätzliches

Vor nicht allzu langer Zeit stand noch im Impressum der Mitgliederzeitung, daß sie mit LATEX 2.09 hergestellt wird. Nachdem das nun schon seit einigen Jahren nicht mehr „State of the Art“ ist, sollte auch bei dem Aushängeschild der deutschsprachigen TEX-Gemeinde etwas aktuelleres Verwendung finden. Als ich dann einen Artikel für „Die TEXnische Komödie“ verfaßt habe [4], ist nebenbei eine Sammlung von Makros entstanden, um den Artikel in dem gewünschten Layout bereits bei der Entstehung kontrollieren zu können. Später habe ich diesen Samen zu einer vollständigen Klasse erblühen lassen.

Ich muß noch erwähnen, daß es natürlich bereits vor meinen Bemühungen einen Dokumentenstil gab, mit dem die bisherigen Ausgaben der Mitgliederzeitung hergestellt wurden. Diesen habe ich für die Layout-Parameter der ersten Version der Klasse `dtk.cls` benutzt.

Beginnen wir also mit den Überlegungen, die jeglicher Arbeit vorausgehen sollten. Das Ziel war es, eine neue Dokumentenklasse zu erstellen. Nun ist es aber leider Tatsache, daß eine beträchtliche Menge von LATEX 2.09-Installationen existieren und verwendet werden. Also wäre es sinnvoll, eine Dokumentenklasse zu haben, die auch als Dokumentenstil verwendbar ist. Aus diesem Grund wird im folgenden immer wieder, und eigentlich viel zu häufig, von LATEX 2.09 die Rede sein müssen.

Obwohl es oft vergessen wird, beschreiben LATEX-Dokumente in erster Linie den *logischen* Aufbau und nicht das Aussehen eines fertigen Schriftstücks. In einer Dokumentenklasse werden die logischen Elemente beschrieben, aus denen ein Dokument aufgebaut sein kann. Dazu wird jeweils festgelegt, wie diese im

gesetzten Dokument aussehen sollen. Zuerst müssen wir uns über den logischen Aufbau eines Dokumentes klar sein, das mit der zu erstellenden Dokumentenklasse verarbeitet werden soll. Um nicht das Rad von neuem zu erfinden, liegt es nahe, als Vorlage bestehende Dokumentenklassen zu nehmen.

Als nächstes müssen wir beachten, daß es mindestens zwei Sichten gibt, von denen aus ein Dokument betrachtet werden kann. Da ist erstens die des Autors eines Artikels, der im allgemeinen nur seinen eigenen Beitrag im Auge hat. Auf der anderen Seite gibt es aber auch noch die der Redaktion, die aus mehreren Artikeln und weiteren Versatzstücken eine komplette Ausgabe zusammensetzen muß.

Für die Autoren ist es am einfachsten, wenn sie die bekannten Befehle der Dokumentenklassen `article`, `report` oder `book` benutzen können. Für sie werden wir nur wenige veränderte oder gar neue Befehle zur Verfügung stellen – auch wenn sich das Layout beträchtlich von dem Layout der Standardklassen unterscheidet. Zur Produktion der gesamten Ausgabe werden allerdings einige zusätzliche Befehle benötigt, um das Leben der Redaktion nicht unnötig zu erschweren.

Zusätzlich gibt es noch den Standpunkt desjenigen, der die neue Klasse schreibt – d. h. meiner. Für den Klassenautor sollte es nicht nötig sein, allzu viel Neues zu erfinden. Dazu ist die Menge von Klassen und Paketen, die jeweils Teilaufgaben lösen, zu groß. Also steht als nächster Schritt eine Sichtung der benötigten Funktionalität und derjenigen Kandidaten von Klassen und Paketen an, die etwas geeignetes beisteuern können.

### Eine Klasse entsteht

Wie wir bereits gesehen haben, bietet es sich an, eine der Standardklassen als Grundlage der Entwicklung zu nehmen. Einzelne Artikel sollen als Teile des Gesamtdokumentes gesetzt werden können, also kommen die Klassen `report` und `book` in Frage. Da „Die TEXnische Komödie“ zweiseitig gesetzt wird, nehmen wir die Klasse `book`.

Als nächstes untersuchen wir, welche Pakete dazugeladen werden sollen. Da „Die TEXnische Komödie“ eine deutschsprachige Publikation ist, werden wir das Paket `german` benutzen. Für die endgültige Zusammenstellung wird außerdem das Paket `multicol` verwendet, das allerdings nur bei Bedarf geladen wird. Damit haben wir die Grundbestandteile der neuen Klasse beisammen.

Nun müssen wir in den saueren Apfel beißen und die weitere Arbeit selbst in die Hand nehmen. Also fangen wir damit an, einige Abkürzungen und Logos





| | |
|---|---|
| `\LaTeX` | LaTeX |
| `\LaTeXe` | LaTeX 2ε |
| `\BibTeX` | BibTeX |
| `\SliTeX` | SliTeX |
| `\MF` | METAFONT |
| `\AmS` | $\mathcal{AMS}$ |
| `\AmSTeX` | $\mathcal{AMS}$-TeX |
| `\LAmSTeX` | LaTeX $\mathcal{AMS}$-TeX |
| `\NTS` | $\mathcal{NTS}$ |
| `\PiCTeX` | PiCTeX |
| `\eV` | e.V. |
| `\dante` | DANTE e.V. |
| `\Dante` | DANTE, Deutschsprachige Anwendervereinigung TeX e.V. |
| `\DTK` | Die TEXnische Komödie |
| `\NFSS` | NFSS |
| `\PS` | PostScript |

Tabelle 1: TEX-Logos und Abkürzungen in `dtk.cls`

zu definieren, die nicht von LaTeX mitgeliefert werden. Dabei benutzen wir die Definitionen aus `texnames.sty` von Nelson H. F. Beebe. Da dieses Paket nicht sehr weit verbreitet ist, wurden die Definitionen in die Klasse mit aufgenommen. Die Liste der wichtigsten in `dtk.cls` bereitgestellten Logos und Abkürzungen ist in Tabelle 1 zu sehen. Wahrscheinlich fehlt noch das eine oder andere, ich wäre deshalb für Hinweise auf solche fehlenden Logos dankbar.

### Der Anfang eines Artikels

Als nächstes sehen wir uns an, wie ein normaler Artikel beginnt. Typischerweise kommt zuerst die Deklaration des Titels mit `\title` und des Autors oder der Autoren mit `\author`, gefolgt von der Anweisung `\maketitle`. An dieser Stelle sind die ersten kleinen Modifikationen nötig. Statt der vollständigen Adresse, die typischerweise im Argument von `\author` angegeben werden kann, wollen wir hier nur den Autorennamen selbst. Die Adressen müssen separat angegeben werden, da diese in der Liste der Autoren am Ende der Mitgliederzeitung zusammengefaßt werden.

Zu diesem Zweck definieren wir einen neuen Befehl `\address`, der drei Argumente hat. Die ersten beiden Argumente zusammen bilden den Namen *eines* Autors. Dabei gibt das zweite Argument an, an welcher Stelle der Autor später alphabetisch eingeordnet werden soll. Das dritte Argument enthält schließlich

die eigentliche Adresse. Ist ein Artikel von mehreren Autoren verfaßt, so wird die Adresse jedes einzelnen Autors angegeben.

Das war jetzt ziemlich theoretisch, sehen wir uns nun ein Beispiel an. Dieser Artikel wurde folgendermaßen eingeleitet:

```
\title{Eine Klasse f"ur "`\DTK"'}
\author{Gerd Neugebauer}
\address{Gerd}{Neugebauer}%
        {Mainzer Str. 8\\
         56321 Rhens\\
         \texttt{gerd@informatik.uni-koblenz.de}}
\maketitle
```

Jeder Artikel sollte durch eine Zusammenfassung, ein sogenanntes *abstract*, eingeleitet werden, durch die die Leser eine Vorstellung vom Inhalt des Artikels erhalten. Das ist eine Gepflogenheit, die sich im wissenschaftlichen Bereich eingebürgert hat.

Um die Zusammenfassung auszeichnen zu können, stellen wir eine Umgebung namens `abstract` zur Verfügung. Der Text der Zusammenfassung wird einfach von dieser Umgebung eingeschlossen. Für diesen Artikel sieht dies folgendermaßen aus:

```
\begin{abstract}
  "`\DTK"' wird neuerdings mit \LaTeXe{} hergestellt...
\end{abstract}
```

### Besonderheiten

Beim Verfassen eines Artikels sind einige Dinge zu beachten. Das meiste versteht sich beinahe von selbst, jedoch werde ich hier auf einige besondere Punkte eingehen.

Wenn wir uns diesen Artikel ansehen, dann fällt auf, daß die Abschnitte nicht numeriert sind, wie dies in den Standardklassen `article`, `report` und `book` der Fall ist. Daher macht es keinen Sinn, ein `\label` an einen Abschnitt (`\section`) anzubringen und dann mit `\ref` auf dessen Nummer zuzugreifen. Es ist jedoch möglich, mit `\pageref` auf die entsprechende Seite zu verweisen.

Autoren müssen auf jeden Fall darauf verzichten, Layout-Parameter zu ändern. Falls dies in einem Artikel unumgänglich erscheint, müssen diese Änderungen lokal auf den Artikel beschränkt werden.





*Literatur*

Ebenfalls eine besondere Rolle spielt die Literatur, die in einem Artikel zitiert wird. Natürlich verwenden wir BIBTEX, um die Literaturangaben zu formatieren. Zu der Klasse `dtk.cls` gibt es einen speziellen BIBTEX-Stil `dtk.bst`, in dem das vorgegebene Layout der Mitgliederzeitung definiert ist.

Da die Literaturhinweise einzeln an jeden Beitrag angehängt werden soll, bedarf es einer einfachen Konvention, die es uns erlaubt, dies ohne viel Aufwand zu bewerkstelligen. Diese Konvention besagt, daß die Literatur zu einem Beitrag in einer einzelnen BIBTEX-Datei enthalten sein muß, die den gleichen Basisnamen trägt wie der eigentliche Beitrag. Wenn z. B. der Artikel in der Datei `ein-hack.tex` steht, muß die zugehörige Literatur in der Datei `ein-hack.bib` zu finden sein.

Die Anwendung von BIBTEX soll hier nicht beschrieben werden. Denjenigen, die noch nicht mit BIBTEX vertraut sind, sei die Lektüre einer Einführung in BIBTEX empfohlen, wie sie in jedem (guten) Einführungsbuch über LATEX zu finden ist (siehe beispielsweise [2, 3]).

Der zur `dtk`-Klasse gehörende BIBTEX-Stil `dtk.bst` wurde mit dem Paket `custom-bib`[1] von Patrick W. Daly erstellt (siehe auch [1, Abschnitt 13.9]). Dieses Paket erlaubt es, anhand von vielen Wahlmöglichkeiten, interaktiv einen BIBTEX-Stil zusammenzustellen, der persönlichen Anforderungen entspricht.

Im BIBTEX-Stil `dtk.bst` wurden, zusätzlich zu den Journalen, die in den Stilen `plain`, `alpha`, etc. vorhanden sind, die folgenden Zeichenketten vordefiniert und müssen deshalb nicht mehr extra vereinbart werden:

| | |
|---|---|
| `DTK` | Die TEXnische Komödie |
| `TUGboat` | TUGboat |
| `TTN` | TEX and TUG News |

So kann ein Artikel, der in der Mitgliederzeitung erschienen ist, folgendermaßen zitiert werden:

```
@Article{neugebauer:klasse,
  author  = {Gerd Neugebauer},
  title   = {Eine Klasse f{\"u}r "`\DTK",
  journal = DTK,
  year    = 1996,
  volume  = {4/95},
  pages   = {6--15}
}
```

[1] Zu finden im CTAN unter `tex-archive/macros/latex/supported/custom-bib`.

### Die Herstellung einer gesamten Ausgabe

Die Herstellung einer gesamten Ausgabe der Mitgliederzeitung entspricht im wesentlichen der Zusammenstellung eines Buches aus einzelnen Artikeln. Das kennen wir von der Klasse `book`: Ein damit gesetztes Dokument kann einzelne Teile enthalten, die mit der Klasse `article` vorbereitet wurden.

Genau dieses Vorgehen liegt auch der Klasse `dtk` zugrunde. Das heißt, es stehen einige Befehle zur Verfügung, die nicht für die einzelnen Beiträge von Belang sind, sondern nur für die Herstellung der gesamten Ausgabe benötigt werden.

Als erstes werden weitere Gliederungsebenen benötigt, da in einem Beitrag bereits `\section`, `\subsection`, etc. Verwendung finden. Die höchste Gliederungsebene ist der Titel des jeweiligen Beitrags selbst. Nun brauchen wir aber Gliederungsebenen darüber, die beispielsweise mehrere Beiträge zusammenfassen. In Anlehnung an das Vorgehen in der Klasse `book` verwenden wir hierzu die Befehle `\chapter` und `\part`.

Das Aussehen der Titelseite ist in einem einzigen Befehl realisiert: die Anweisung `\DieTeXnischeKomoedie` erzeugt eine komplette Titelseite. Die drei benötigten Argumente geben jeweils die Nummer der Ausgabe, den Jahrgang und das Erscheinungsdatum an.

Für einige Listen – wie den „Spielplan“ oder die Stammtischtermine – hat es sich als nützlich erwiesen, eine spezielle Umgebung zur Verfügung zu stellen. Die Umgebung `roll` entspricht in etwa der Umgebung `description`. Nur ist es hier möglich, die Einrücktiefe als optionales Argument an die Umgebung mitzugeben. Die Wirkung ist an dem folgenden Beispiel zu sehen:

```
\begin{roll}[0pt]
  \item[69008] Heidelberg\\
    Luzia Dietsche\\
    Tel.: ...
\end{roll}
```

**69008** Heidelberg
Luzia Dietsche
Tel.: ...

Diese Umgebung ist natürlich nicht auf die Anwendung in den genannten Fällen beschränkt, sondern könnte unter Umständen auch in den einzelnen Beiträgen Anwendung finden.

Als nächstes kommen wir auf einen Punkt zurück, der bereits angesprochen wurde. Am Anfang haben wir gesehen, wie die Adresse eines Autors in einem Beitrag angegeben werden sollte. Wenn wir eine fertige Ausgabe durchblättern, können wir erahnen, was daraus wird – nämlich die sortierte Liste der Autoren einer Ausgabe. Auch hierfür haben wir eine eigene Umgebung bereitgestellt.





Da die Umgebung und ihr Inhalt, die Liste der Autoren, automatisch erzeugt werden, also weder von den Autoren noch von der Redaktion je „von Hand“ eingegeben werden, ist ihre Benutzung nicht von allgemeinem Interesse und ich werde nicht weiter darauf eingehen.

Bemerkt werden soll aber noch das Folgende: Da die Liste sortiert ausgegeben werden soll, stellte sich die Frage, wie dies am einfachsten zu bewerkstelligen ist. Auf Anhieb ergeben sich hier mehrere Möglichkeiten. Das Sortieren kann von Hand geschehen – die manuelle Lösung. Das Sortieren kann von TEX erledigt werden – die „TEXnische“ Lösung. Oder das Sortieren kann durch ein spezielles Programm erfolgen – die „externe“ Lösung. Ich habe mich für die externe Lösung entschieden. Das Programm, das dabei zum Einsatz kommt, ist natürlich nicht irgendein Sortierprogramm, sondern gehört zur TEX-Familie, nämlich *MakeIndex*.

Um die Aufgabe des Sortierens und Aufbereitens der Liste der Autoren zu erledigen, werden die gewünschten Informationen mit dem bereits beschriebenen Befehl `\address` in eine Datei geschrieben. Damit *MakeIndex* das Sortieren und Formatieren erledigen kann, wurden mit einer Index-Stildatei seine Parameter angepaßt. Nach dem Aufruf von *MakeIndex* wird die fertige Liste beim darauf folgenden LATEX-Lauf eingebunden, wie es auch beim Literaturverzeichnis, Inhaltsverzeichnis, etc. der Fall ist.

Wir können aus dieser Anwendung die Lehre ziehen, daß die Programme der TEX-Familie häufig zu mehr zu gebrauchen sind, als ihr Name andeutet. Ein anderes Beispiel dafür ist BIBTEX, das vielfältiger einsetzbar ist denn für pure Literaturverwaltung. Doch dazu vielleicht ein anderes Mal mehr.

### Die Realisierung

In diesem Abschnitt werde ich einige Probleme und Lösungen vorstellen, wie sie im Zusammenhang mit der Klasse vorkommen. Hierzu kommen wir zu der dritten Sichtweise – nämlich der des Klassenautors. Wie wir gesehen haben, wollen wir für `dtk.cls` größtenteils die logische Sichtweise aus den Klassen `article` und `book` beibehalten.

Also `book.cls` einbinden, ein paar Makros umdefinieren, ein paar neue Makros dazu und fertig. Doch halt! So einfach können wir es uns nicht machen. Schließlich gilt es auch noch, eine weitere Randbedingung zu erfüllen: die Klasse soll gleichzeitig als Dokumentenstil verwendbar sein – für diejenigen, die immer noch LATEX 2.09 verwenden.

Dadurch wird die Realisierung doch etwas komplizierter. Als erstes müssen wir feststellen, ob mit LaTeX 2.09 oder LaTeX $2_\varepsilon$ gearbeitet wird. Das geschieht dadurch, daß wir überprüfen, ob das Makro `\documentclass` definiert ist. Dieser Test scheint ausreichend dafür zu sein, die beiden Versionen voneinander unterscheiden zu können. Das Ergebnis wird dazu benutzt, eine neue Bedingung zu formulieren. So steht am Anfang von `dtk.cls`

```
\newif\ifOldLaTeX
\@ifundefined{documentclass}{\OldLaTeXtrue}{\OldLaTeXfalse}
```

Nun können wir mit dieser neuen Bedingung abprüfen, ob `dtk` als Dokumentenklasse oder als Dokumentenstil benutzt wird. Entsprechend müssen wir verschiedene Vorbereitungen treffen. Das sieht folgendermaßen aus:

```
\ifOldLaTeX
  ...
\else
  ...
\fi
```

Die ersten drei Auslassungspunkte stehen dabei für Definitionen von Makros, die von LaTeX $2_\varepsilon$ bereitgestellt werden, aber in LaTeX 2.09 nicht vorhanden sind. Diese werden hier nachgebildet und können im folgenden benutzt werden, ohne daß diese Fallunterscheidung noch einmal durchgeführt werden muß.

Die zweiten Auslassungspunkte stehen für den üblichen Vorspann zur Definition einer Klasse unter LaTeX $2_\varepsilon$:

```
\ProvidesClass{dtk}[1996/02/21 gene]
\DeclareOption{full}{\DTK@Author@false}
\DeclareOption{10pt}{\OptionNotUsed}
\DeclareOption{11pt}{\OptionNotUsed}
\DeclareOption{12pt}{\OptionNotUsed}
\DeclareOption{twoside}{\OptionNotUsed}
\DeclareOption{oneside}{\OptionNotUsed}
\DeclareOption*{\PassOptionsToClass{\CurrentOption}{book}}
\ProcessOptions
```

Die Klasse wird mit dem Referenzdatum bekannt gemacht, die meisten Optionen der Klasse `book` werden ausgeschaltet und die restlichen Optionen an die Klasse `book` weitergegeben. Als einzige Option wird die `full` von der Klasse selbst ausgewertet. Sie ist dazu vorgesehen, die Version der Autoren, einen einzelnen Beitrag, von der Vollversion, der gesamten Ausgabe, zu unterscheiden. Für die Vollversion laden wir später zusätzlich das Paket `multicol` nach.





Schließlich müssen wir noch die Klasse `book` laden, dies geschieht mit dem Befehl `\LoadClass`, und das Paket `german` mit dem Befehl `\RequirePackage`.

Wir überspringen nun etliche Einstellungen von Layout-Parametern, Definitionen von neuen und Redefinitionen von alten Makros. Als nächsten interessanten Punkt wollen wir uns die Definition der Umgebung `roll` ansehen. In LaTeX $2_\varepsilon$ könnten wir sehr einfach eine Umgebung mit einem optionalen Argument definieren. Das ist in LaTeX 2.09 nicht so einfach möglich, daher können wir nicht die erweiterten Definitionsmöglichkeiten aus LaTeX $2_\varepsilon$ verwenden, sondern müssen uns auf die Mittel beschränken, die LaTeX 2.09 bietet.

Zuerst wird die neue Umgebung `roll` definiert. Am Anfang wird überprüft, ob ein optionales Argument vorhanden ist, falls nicht, wird ein Standardwert übergeben. Am Ende wird die `list`-Umgebung geschlossen, die in `\r@ll`, der zweiten Definition, begonnen wurde.

```
\newenvironment{roll}%
  {\@ifnextchar[{\r@ll}{\r@ll[0pt]}}%
  {\end{list}}
```

Die zweite Definition benutzt die Umgebung `list`, bei deren Aufruf einige Parameter eingestellt werden. Diese Definition muß mit dem TEX-Befehl `\def` erfolgen, da das Argument von eckigen Klammern eingeschlossen sein soll. Das läßt sich ansonsten mit LaTeX-Befehlen nicht realisieren.

```
\def\r@ll[#1]{\begin{list}{}{%
    \labelwidth   #1
    \leftmargin   \labelwidth
    \itemsep      .3ex
    \let\makelabel\rolllabel}}
```

Schließlich bestimmt eine dritte Definition, wie das Label der Liste formatiert werden soll:

```
\newcommand\rolllabel[1]{\hspace\labelsep\bf #1\hfill}
```

Als nächstes sehen wir uns an, wie die Liste der Adressen in „Die TEXnische Komödie“ kommt. Wie wir bereits gesehen haben, wird diese Liste von *MakeIndex* in eine Datei mit der Endung `.ind` geschrieben. Also müssen wir nur nachsehen, ob es eine solche Datei gibt und sie gegebenenfalls einlesen. Anderenfalls geben wir einen Hinweis aus, wie solch eine Datei zu erstellen ist:

```
\newcommand\listofaddresses{%
  \InputIfFileExists{\jobname.ind}{}{%
    \typeout{No file \jobname.ind.}%
    \typeout{Use the following command to create it:\space\space
      makeindex -s dtk.ist \jobname}%
    \typeout{\space}}}
```

Mit diesem kurzen Abriß wollen wir es genug sein lassen. Die neue Klasse enthält natürlich noch weitere Definitionen, beispielsweise die Realisierung des Layouts der Titelseite oder den Mechanismus, über den die einzelnen Adressen in die Indexdatei geschrieben werden, allerdings würde es zu weit führen, all dies hier zu beschreiben. Wer sich für die Lösung dieser Probleme interessiert, sei an den dokumentierten Quellcode verwiesen.

### Schlußbemerkung

Die aktuelle Version der Klasse `dtk` muß im Augenblick noch als Beta-Version betrachtet werden. Wer sich trotzdem dafür interessiert, kann sie über den Autor oder über DANTE e.V. bekommen. In naher Zukunft wird `dtk` auch auf CTAN zu finden sein. Mit dieser neuen Klasse sollte es ein Vergnügen sein, einen Artikel für „Die TEXnische Komödie“ zu schreiben. Also ran!

### Literatur

[1] Michel Goosens, Frank Mittelbach und Alexander Samarin: *The LATEX Companion*; Addison-Wesley Publishing Company, Reading, Mass., 1994.

[2] Helmut Kopka: *LATEX – Eine Einführung*; Addison-Wesley Publishing Company, Bonn, 2. Aufl., 1996.

[3] Leslie Lamport: *LATEX – A Document Preparation System*; Addison-Wesley Publishing Company, Reading, Mass., 2. Aufl., 1994.

[4] Gerd Neugebauer: BibTool – *Manipulation von* BibTEX*-Dateien*; in: *Die TEXnische Komödie*, 4/94, S. 4–10, 1994.



## Orale Spielereien mit TEX – Teil III

Bernd Raichle

Schon gemerkt, TEX fügt manchmal von sich aus einfach ein oder mehrere Tokens in die Eingabe ein? Sie wissen nicht, worüber ich schreibe? Probieren Sie



es doch einfach einmal aus. Geben Sie in einem Dokument das Zeichen ^ ein und TEX sagt Ihnen mit der Fehlermeldung „`Missing $ inserted.`“, daß er das Token `$` eingefügt hat. Dieses Verhalten von TEX dürfte Ihnen sicherlich schon bekannt sein. Jedoch gibt es auch einige Situationen, in denen TEX unbemerkt und ohne einen Fehler ein Token in die Eingabe einfügt.

Beispielsweise will ein Benutzer am Ende eines Dokuments die Anzahl der erzeugten Seiten in der Form „*n* Seiten“ in eine Datei schreiben. Dabei soll für den Spezialfall „1 Seite“ das Plural-n weggelassen werden. Die Lösung sieht im Prinzip wie folgt aus:

```
\countdef\SeitenNr=0  \SeitenNr=1
\message{\number\SeitenNr\space Seite\ifnum\SeitenNr=1\else n\fi.}
```

Leider taucht danach in der Ausgabe unerwartet ein `\relax` auf:

```
1 Seite\relax .
```

Nach langem Herumprobieren findet unser armer Benutzer dann mit

```
\message{\number\SeitenNr\space Seite\ifnum\SeitenNr=1 \else n\fi.}
```

eine Lösung, die ohne das unerwünschte `\relax` funktioniert. Der Ihnen vorliegende Teil III dieser Serie geht daher etwas näher darauf ein, wann und wieso TEX dieses `\relax` einfügt, und wie man es vermeiden oder nutzen kann. Die dabei gezeigten Techniken und Tricks sind sehr speziell und nur dann vonnöten, wenn Ihre Makros vollständig expandieren müssen, ohne daß in manchen Fällen ein störendes `\relax` übrigbleiben wird.

### Woher kommt das eingefügte Token `\relax`?

Woher kommt nun das Token `\relax` und wieso fügt es TEX ein? Die Antwort auf diese Fragen wird durch einen Blick auf die Situationen augenfällig, in denen TEX dieses `\relax` durch die Prozedur *insert_relax* [2, §379] einfügt.

- Liest TEX nach den Anweisungen `\input`, `\font`, `\openin` und `\openout` einen Dateinamen, und taucht in der Eingabe, bevor der Dateiname abgeschlossen wurde, die Anweisung `\input` auf, so wird vor diesem `\input` ein `\relax` eingefügt [2, §378]. Dieses `\relax` beendet den Dateinamen, so daß die vorherige Anweisung ausgeführt werden kann, bevor TEX mit der Bearbeitung der neuen `\input`-Anweisung beginnen wird. Die Anweisung `\input` ist expandierbar; wäre sie das nicht, würde man das eingefügte `\relax` nicht benötigen.

- In *if*-Abfragen fügt TEX ein `\relax` jedesmal dann ein, wenn ein `\fi`, `\else` oder `\or` eingelesen wird, bevor die Bedingung beendet und ausgewertet werden kann [2, §510]. Wie im vorherigen Fall mit `\input` sind auch hier die Tokens `\fi`, `\else` bzw. `\or` expandierbar und das eingefügte Token `\relax` beendet die Bedingung.

In beiden Situationen dient das Token `\relax` dazu, die Auswertung einer Anweisung zu beenden, bevor weitere Tokens abgearbeitet werden, deren Auswertung wiederum vom Ergebnis dieser Anweisung abhängen. Außerdem wird ein `\relax` in diesen Situationen nur vor den Tokens `\input`, `\fi`, `\else` oder `\or` eingefügt, die allesamt expandierbar sind.

**Wie vermeide ich das eingefügte Token `\relax`?**

Beim `\input` stört es in den wenigsten Fällen, daß TEX in der genannten Situation ein `\relax` vor diesem einfügt. Im Grunde genommen ist es nicht notwendig, daß die Anweisung `\input` expandierbar ist. Diese Eigenschaft wird bisher von keinem Makropaket genutzt. In LATEX wird beispielsweise das Primitiv `\input` durch ein Makro gleichen Namens überschrieben, und diese neue Definition ist nicht mehr vollständig expandierbar. Dies wurde bisher nicht als Beschränkung des „neuen“ `\input` angesehen.

Im Gegensatz dazu stört bei *if*-Abfragen das eingefügte Token `\relax`, wenn man ein vollständig expandierbares Makro definieren muß. Die *if*-Abfragen, bei denen das erwähnte `\relax` stören könnte, kann man anhand der für die Bedingung gelesenen Datentypen in drei Klassen einteilen:

1. Bedingungen, die eine oder zwei ganze Zahlen ⟨number$_{1,2}$⟩ vergleichen,
2. Bedingungen, die zwei Längen ⟨dimen$_{1,2}$⟩ vergleichen,
3. Bedingungen, die zwei einzelne Tokens ⟨token$_{1,2}$⟩ vergleichen.

In allen Fällen versucht TEX zum Lesen der Zahlen, der Längen bzw. der beiden Tokens alle Tokens zu expandieren, bis eine für einige Bedingungen notwendige Vergleichsrelation und die geforderten Operanden vorliegen.

*Vergleich von ganzen Zahlen*

In den folgenden *if*-Abfragen werden ganze Zahlen als Operanden benötigt:

- `\ifnum`⟨number$_1$⟩⟨relation⟩⟨number$_2$⟩ (vergleiche zwei ganze Zahlen)
- `\ifodd`⟨number⟩ (teste auf eine ungerade ganze Zahl)
- `\ifvoid`⟨number⟩, `\ifhbox`⟨number⟩, `\ifvbox`⟨number⟩ (teste auf Inhalt und Typ des angegebenen Box-Registers)





- `\ifeof`⟨number⟩ (teste auf das Ende einer Datei)
- `\ifcase`⟨number⟩⟨text for case 0⟩`\or` ... (mehrfache Verzweigung)

Im folgenden werde ich mich auf die Abfrage mit `\ifnum` und den Vergleich auf Gleichheit der beiden Zahlen beschränken. Für die anderen Abfragen und Relationen gelten grundsätzlich dieselben Aussagen.

Eine ganze Zahl kann direkt durch Vorzeichen und einzelne Ziffer-Tokens eingegeben werden. Eine so gebildete Zahl wird entweder durch ein Leerzeichen oder ein Token, das nicht expandierbar ist und keiner Ziffer entspricht, beendet.

```
\message{-\ifnum 12=12 \fi-}                    % Ausgabe: --
\message{-\ifnum 12=12a\fi- -\ifnum 11=12a\fi-} % Ausgabe: -a- --
```

Wie man anhand der ersten Zeile dieses Beispiels sieht, ist ein Leerzeichen, mit dem die Zahl beendet werden soll, Bestandteil der Zahl und wird daher nicht mit ausgegeben. Nicht expandierbare Tokens, die wie das `a` in der zweiten Zeile des Beispiels die Zahl beenden, gehören nicht mehr zur Zahl, sondern sind schon Bestandteil des *then*-Teils der Bedingung. Entfernen wir nun das Leerzeichen bzw. das nicht expandierbare Token `a` in unserem Beispiel.

```
\message{-\ifnum 12=12\fi-}             % Ausgabe: -\relax -
```

Wie man erkennen kann, ist das Leerzeichen oder ein anderes Token, das eine Zahl beendet, in diesem Fall notwendig, um ein von TEX automatisch eingefügtes `\relax` zu verhindern. Was hindert uns also daran, in einem verallgemeinerten Makro, in dem wir eine Zahl vergleichen wollen, einfach dieses Leerzeichen als Ende der Zahl mit einzugeben? Statt

```
\def\VergleicheZahlMitNull#1{%
  \message{-\ifnum 0=#1\fi-}}
```

müßte doch ein Leerzeichen nach `#1`, wie in

```
\def\VergleicheZahlMitNull#1{%
  \message{-\ifnum 0=#1 \fi-}}
```

das eventuell von TEX eingefügte `\relax` vermeiden?

Leider funktioniert dies nicht in allen Fällen. Solange die Zahl im ersten Argument immer durch einzelne Ziffern angegeben wird, funktioniert unser kleiner Trick.

```
\VergleicheZahlMitNull{0}               % Ausgabe: --
\VergleicheZahlMitNull{00}              % Ausgabe: --
\VergleicheZahlMitNull{1}               % Ausgabe: --
```

Jedoch kann man in TEX eine ganze Zahl auch als sogenannte interne Zahl ⟨internal integer⟩ angeben. Ein solche interne Zahl kann beispielsweise ein Parameter in TEX, wie `\day` oder `\widowpenalty`, oder ein durch `\newcount` (oder `\newcounter` in LATEX) definierter und durch eine *control sequence* benannter Zähler sein. Hierbei wird für die Zahl selbst nur ein einziges Token gelesen, eben die *control sequence* als Name des Parameters oder des Zählers. TEX versucht danach nicht, weitere Tokens für diesen Operanden zu lesen. Vergleicht man also mit solch einer Zahl,

```
\newcount\Null \Null=0  \day=0  \chardef\NullChar=0
\VergleicheZahlMitNull{\Null}          % Ausgabe: - -
\VergleicheZahlMitNull{\day}           % Ausgabe: - -
\VergleicheZahlMitNull{\NullChar}      % Ausgabe: - -
```

bleibt leider nach der Expansion unser zusätzlich eingefügtes Leerzeichen übrig und ist dann Bestandteil des *then*-Teils der Bedingung. Um aus diesem Dilemma herauszufinden, muß man sich also doch etwas Komplizierteres einfallen lassen.

Eine sehr einfache Lösung bietet sich für `\ifnum` immer dann an, wenn eine der beiden Zahlen konstant ist oder vorher bekannt ist, ob die Zahl als interne Zahl oder durch einzelne Ziffern vorliegt. In diesem Fall wissen wir vorher, ob ein Leerzeichen zum Beenden dieser Zahl notwendig ist oder nicht, und sie wird, mit oder ohne Leerzeichen, auf der *rechten* Seite der Relation plaziert. Auf der linken Seite, also vor der Relation, wird die über das Argument angegebene Zahl gesetzt. Durch das Token für die Vergleichsrelation, das nicht expandierbar ist, wird diese Zahl beendet, so daß wir nie ein zusätzliches Leerzeichen benötigen.

```
\def\VergleicheZahlMitNull#1{%
  \message{-\ifnum #1=0 \fi-}}
\newcount\Null \Null=0 % ...oder andere Konstante
\def\VergleicheZahlMitNull#1{%
  \message{-\ifnum #1=\Null\fi-}}
```

Leider funktioniert dieser Trick nur für `\ifnum` und hier auch nur für den Fall, daß von einem der beiden Operanden vorher bekannt ist, ob er als interne Zahl oder als Einzelziffern vorliegt. Für die anderen *if*-Abfragen muß man sich daher etwas anderes einfallen lassen, falls das zusätzlich eingefügte `\relax` stören sollte.

Überlegen wir uns noch einmal, was erforderlich ist: Auf der einen Seite benötigt man ein nicht expandierbares Token, um eine aus Ziffern bestehende Zahl zu beenden. Auf der anderen Seite sollte dieses Token auch wieder zu





eine leeren Folge von Tokens expandieren, da wir sonst gleich das eventuell eingefügte `\relax` beibehalten könnten. Beide Anforderungen hören sich zuerst widersprüchlich an. Das Problem kann aber mit `\noexpand\empty` gelöst werden. Das Primitiv `\noexpand` führt dazu, daß das expandierbare Token `\empty`, ein zu einer leeren Folge von Tokens expandierendes Makro, bei der nächsten Expansion temporär die Bedeutung eines `\relax` besitzt und daher nicht expandiert wird.

Unsere neue Definition von `\VergleicheZahlMitNull`

```
\def\VergleicheZahlMitNull#1{%
  \message{-\ifnum 0=#1\noexpand\empty\fi-}}
```

funktioniert leider wiederum nicht in allen Fällen, da ja das `\noexpand` das nachfolgende, normalerweise expandierbare Token `\empty` zu einem nicht expandierbaren Token macht. Daher verschwindet dieses Token nicht, wenn man in der Bedingung eine interne Zahl angibt.

```
\VergleicheZahlMitNull{0}              % Ausgabe: --
\VergleicheZahlMitNull{\Null}          % Ausgabe: -\empty -
```

Da ein `\noexpand` das nachfolgende Token nur *einmalig* vor Expansion schützt, können wir diesen Trick mit `\noexpand\empty` dennoch überall anwenden, wo die Expansion der Bedingung später erneut expandiert wird. Das nachfolgende Beispiel zeigt dies.

```
\def\VergleicheZahlMitNull#1{%
  \edef\text{-\ifnum 0=#1\noexpand\empty\fi-}%
  \message{\text}}
\VergleicheZahlMitNull{0}              % Ausgabe: --
\VergleicheZahlMitNull{\Null}          % Ausgabe: --
```

Leider fehlt jetzt noch eine mögliche Lösung für den Fall, daß die Bedingung nur ein einziges Mal expandiert wird. Hier kann man, wie gesehen, `\noexpand` nicht verwenden, da dann in einigen Fällen das folgende Token nicht mehr expandiert wird. Es bleibt damit nichts anderes übrig, als ein eventuell eingefügtes `\relax` nach dem Auswerten der Bedingung wieder „von Hand“ zu entfernen und dann fortzufahren.

```
\def\CheckForRelax#1{%
  \ifx\relax#1%   % Token = \relax?
    \expandafter\ForgetArgument
    % => \ForgetArgument{#1}
  \else
    \expandafter\DoNotForgetArgument
```

```
    % => \DoNotForgetArgument{#1}
  \fi {#1}}
\def\ForgetArgument#1{}
\def\DoNotForgetArgument#1{#1}
```

Die seltsam anmutende Plazierung des Arguments `#1` der Makros `\Forget-Argument` und `\DoNotForgetArgument` am Ende des Makros `\CheckForRelax` ist notwendig, da auch ein `\else`, `\or` oder `\fi` in diesem Argument enthalten sein kann. Da *if-fi*-Abfragen korrekt geschachtelt sein müssen, darf aus diesem Grund das Argument erst nach dem `\fi` wieder eingefügt werden.

Mit Hilfe des Makros `\CheckForRelax` wird ein eventuell eingefügtes `\relax` auch bei einmaliger Expansion entfernt.

```
\def\VergleicheZahlMitNull#1{%
  \message{-\expandafter\CheckForRelax\ifnum 0=#1\fi-}}
\VergleicheZahlMitNull{0}               % Ausgabe: --
\VergleicheZahlMitNull{\Null}           % Ausgabe: --
```

Wie so oft gibt es jedoch auch hier noch einige Einschränkungen, da das Makro `\CheckForRelax` immer ein Token liest, auch wenn es Bestandteil des *then*- oder *else*-Teils der Bedingung ist. Ist dieses Token ein `\relax`, so wird es entfernt, auch wenn es aus guten Gründen vom Makroprogrammierer an diese Stelle plaziert wurde. Außerdem kann es beispielsweise zu Problemen kommen, wenn das erste Token statt eines einzelnen Tokens eine Tokenliste ist. In diesen Fällen wird die Klammerung entfernt, die Tokenliste also als Folge einzelner Tokens wieder in die Eingabe gestellt.

Will man auch mit dieser Einschränkung nicht leben, bleibt nur die Lösung, die Tokens im Argument vor dem Vergleich daraufhin abzuprüfen, ob die Zahl als interne Zahl oder mittels einzelner Ziffern unserem Makro `\VergleicheZahl-MitNull` übergeben wurde. Da wiederum die Ziffern in Makros „versteckt“ sein könnten, muß man dazu zuerst diese Makros zu einer Zahl und diese Zahl mit dem Primitiv `\number` zu einzelnen Ziffern expandieren, bevor die Ziffern mit einem Leerzeichen beendet erneut von TEX als Zahl gelesen werden. Da auch `\number` eine Zahl aus einzelnen Ziffern mit einem optionalen Leerzeichen liest, hat man auch hier wiederum das Problem, daß man zuvor nicht weiß, wieviele Leerzeichen einzufügen sind, um das `\relax` zu vermeiden.

```
\def\VergleicheZahlMitNull#1{%
  \message{-\ifnum 0=\number#1 \fi-}}
\VergleicheZahlMitNull{0}               % Ausgabe: -\relax -
\VergleicheZahlMitNull{\Null}           % Ausgabe: --
```





```
\def\VergleicheZahlMitNull#1{%
  \message{-\ifnum 0=\number#1 \space\fi-}}
\VergleicheZahlMitNull{0}               % Ausgabe: --
\VergleicheZahlMitNull{\Null}           % Ausgabe: - -
```

Damit wäre auch hier eine Lösung notwendig, die in einem Fall ein zusätzliches Leerzeichen einfügt, es im anderen Fall aber unterläßt. Letztlich hat man sich damit nur im Kreis gedreht und kann für eine Lösung auf Seite 18 weiterlesen. Andererseits habe ich ja immer noch die Hoffnung, daß einer von Ihnen, liebe Leser, doch noch eine etwas elegantere Lösung als das vorgestellte `\noexpand \empty` oder `\CheckForRelax` finden könnte oder vielleicht schon gefunden hat.

*Vergleich von Längen*

Für den Vergleich zweier Längen mit

- `\ifdim`⟨dimen$_1$⟩⟨relation⟩⟨dimen$_2$⟩

gelten diesselben Aussagen wie für `\ifnum`. Längenangaben können wie ganze Zahlen entweder durch einzelne Ziffern und eine Einheit oder eine interne Länge angegeben werden. Im ersten Fall sucht TEX nach dem Lesen der Einheit noch nach einem optionalen Leerzeichen, was dazu führen kann, daß auch hier eventuell ein `\relax` eingefügt wird.

```
\newdimen\NullLaenge \NullLaenge=0pt
\message{-\ifnum 0pt=0pt \fi-}          % Ausgabe: --
\message{-\ifnum 0pt=\NullLaenge\fi-}   % Ausgabe: --
\message{-\ifnum 0pt=0pta\fi-}          % Ausgabe: -a-
\message{-\ifnum 0pt=0pt\fi-}           % Ausgabe: -\relax -
```

Auch für Vergleiche von Längen kann man dieselben Tricks wie bei `\ifnum` anwenden, um ein `\relax` zu verhindern. `\CheckForRelax` kann auch hier das `\relax` nachträglich entfernen.

```
\def\VergleicheLaengeMitNull#1{%
  \message{-\expandafter\CheckForRelax\ifdim 0pt=#1\fi-}}
\VergleicheLaengeMitNull{0pt}           % Ausgabe: --
\VergleicheLaengeMitNull{\NullLaenge}   % Ausgabe: --
```

*Vergleich von Tokens*

Die Vergleiche von Tokens

- `\if`⟨token$_1$⟩⟨token$_2$⟩
  (vergleiche, ob die Zeichen-Codes der beiden Tokens übereinstimmen)

- `\ifcat`⟨token$_1$⟩⟨token$_2$⟩
  (vergleiche, ob beide Tokens denselben *category code* besitzen)

laufen etwas anders ab als der Vergleich zweier Zahlen oder Längenangaben. Hier werden grundsätzlich nur zwei Tokens gelesen, es wird also nie ein optionales Leerzeichen nach diesen beiden Tokens gesucht.

```
\message{-\if aa\else|\fi-}          % Ausgabe: --
\message{-\if ab\else|\fi-}          % Ausgabe: -|-
```

Enthält die Bedingung weniger als zwei Tokens, liest also TEX zuvor ein `\else`, `\or` oder `\fi`, werden entsprechend ein bzw. zwei `\relax`-Tokens eingefügt.

```
\message{-\if a\else|\fi-}           % Ausgabe: -|-
\message{-\if \relax\else|\fi-}      % Ausgabe: --
\message{-\if\else|\fi-}             % Ausgabe: --
```

Ein eingefügtes `\relax` wird jedoch für den Vergleich gelesen, es wird also entfernt. Man hat demnach keine Probleme, daß ein eventuell eingefügtes `\relax` in der Expansion auftauchen könnte.

Der Vergleich mit `\ifx` erscheint nicht in der obenstehenden Übersicht, da `\ifx` die nächsten beiden Tokens ohne Expansion liest. Dabei kann auch ein `\else`, `\or`, `\fi` und sogar ein `\if...` gelesen werden, ohne daß TEX dies dadurch verhindert, daß automatisch, genauso wie bei den anderen Vergleichen, ein `\relax` eingefügt wird.

```
\message{-\ifx\fi\fi aha!\fi-}       % Ausgabe: -aha!-
```

**Wie nutze ich das eingefügte Token `\relax`?**

Wie man bisher gesehen hat, will man bei Vergleichen von Zahlen und Längen das `\relax` vermeiden, wenn man vollständig expandierbare Makros benötigt. In diesen Fällen ist also das `\relax` eher hinderlich, da es nicht expandiert.

In vielen Fällen ist ein `\relax` zwingend notwendig, um die Zahl oder Länge eindeutig vom *then*- oder *else*-Teil zu trennen. Beispielsweise muß man manchmal eine vorzeitige Expansion der dort stehenden Makros verhindern. Hier ist es am einfachsten, wenn man ein `\relax` ohne Leerzeichen nach dem zweiten Operanden explizit einfügt.

Bei einem Vergleich von zwei Tokens mit `\if` bzw. `\ifcat` kann aber das automatische Einfügen eines `\relax` für elegante Tricks genutzt werden. Wie schon in den obigen Beispielen für `\if` kann man beispielsweise das `\relax` nutzen, um auf ein optional angegebenes Zeichen zu testen.





```
\def\CheckForA#1{%
  \message{\if A#1\else Kein A\fi.}}
\CheckForA{}                         % Ausgabe: Kein A.
\CheckForA{A}                        % Ausgabe: .
\CheckForA{a}                        % Ausgabe: Kein A.
```

Dieses Makro versagt jedoch spätestens dann, wenn das Argument mehr als ein Zeichen enthält, da die zusätzlichen Zeichen zu einem Bestandteil des *then*-Teils werden.

```
\CheckForA{aB}                       % Ausgabe: Kein A.
\CheckForA{AB}                       % Ausgabe: B.
```

Dennoch führt diese Technik in einigen Situationen zu sehr eleganten, wenn auch für Anfänger eher ungewohnten und dadurch schwer zu durchschauenden Lösungen. Der hier beschriebene Trick stammt von David Kastrup, der ihn in einem Makro namens `\ifgreek` in einer neuen, bisher noch unveröffentlichten Version des Paketes „kdgreek“ verwendet. Folgendes ist eine geringfügig geänderte Variante dieses Makros.

```
\def\ifKGR#1{%
  \iftrue
    \if R\if G\if K#1\fi\fi
    \else
      \expandafter\expandafter\expandafter\iffalse\fi
    \fi}
```

Das Makro vergleicht sein Argument mit der Tokenfolge „K“, „G“, „R“. Dabei darf das Argument weniger, aber nie mehr als drei Zeichen enthalten.

```
\message{\ifKGR{}ja\else nein\fi}       % Ausgabe: nein
\message{\ifKGR{kgr}ja\else nein\fi}    % Ausgabe: nein
\message{\ifKGR{KG}ja\else nein\fi}     % Ausgabe: nein
\message{\ifKGR{KGR}ja\else nein\fi}    % Ausgabe: ja
```

Üblicherweise wird ein Makro, das wie das hier gezeigte Makro `\ifKGR` zwei als Folge von Tokens vorliegende „Zeichenketten“ miteinander vergleicht, wie folgt realisiert:

```
\usualifKGR#1{%
  \def\0{KGR}\edef\1{#1}%
  \ifx\0\1}
```

Im Unterschied zu dem oben gezeigten Makro `\ifKGR` ist `\usualifKGR` durch die Definitionen der beiden Hilfsmakros `\0` und `\1` nicht vollständig expandierbar. Es ist aus diesem Grund *zerbrechlich* und müßte in entsprechenden Situationen in LATEX mit `\protect` geschützt werden.

Das Makro `\ifKGR` besitzt noch eine weitere Besonderheit. Normalerweise ist es bei vollständig expandierbaren Makros schwierig, die Tokens im Argument dieses Makros komplett über mehrere Stufen zu expandieren. Diese vollständige Expansion, die in einigen Situationen notwendig ist, erhält man bei der Verwendung eines `\if` im Unterschied zum nur einmalig expandierenden `\expandafter` automatisch:

```
\def\macroA{KGR}
\def\macroB{\macroA}
\message{\ifKGR{\macroB}ja\else nein\fi}
```

Sehen wir uns `\ifKGR` deshalb etwas genauer an. Am anschaulichsten wird die Funktionsweise dieses Makros an einem Beispiel und den stattfindenden Expansionsschritten für den ersten negativen Fall verdeutlicht:

```
1 \ifKGR{}ja\else nein\fi
2 \iftrue \if R\if G\if K\fi\fi \else ... \fi ja\else nein\fi
```

Das `\iftrue` ist immer wahr. TEX merkt sich für das `\iftrue`, daß noch ein dazu passender *then*- und *else*-Teil folgt, der mit `\else` bzw. abschließend mit `\fi` beendet werden muß.

```
3 \if R\if G\if K\fi\fi \else ... \fi ja\else nein\fi
4 \if R\if G\if K\relax\fi\fi \else ... \fi ja\else nein\fi
```

`\if` expandiert Tokens so lange, bis es zwei nicht expandierbare Tokens vorfindet. Da `\if` wiederum expandierbar ist, expandiert das erste, äußere `\if` das zweite und dieses das dritte, innerste `\if`. Erst hier tritt der Fall auf, daß ein `\fi` gelesen wird, bevor zwei Tokens zum Vergleich gelesen werden konnten und es wird ein `\relax` eingefügt. Dieses `\relax` wird dann mit dem Token „K“ verglichen. Der Vergleich fällt natürlich negativ aus, so daß alles bis zum nächsten `\fi` überlesen wird.

```
5 \if R\if G\fi \else ... \fi ja\else nein\fi
6 \if R\if G\relax\fi \else ... \fi ja\else nein\fi
7 \if R\else ... \fi ja\else nein\fi
8 \if R\relax\else ... \fi ja\else nein\fi
```

Diesselben Schritte werden für die beiden weiteren `\if` abgearbeitet, bis beim ersten, äußeren `\if` festgestellt wird, daß der Vergleich falsch ist. Somit wird der bisher nur mit ... dargestellte *else*-Zweig der Bedingung abgearbeitet.





```
9 \expandafter\expandafter\expandafter\iffalse\fi \fi ja\else nein\fi
10 \expandafter\iffalse \fi ja\else nein\fi
```

Die drei `\expandafter`-Anweisungen sorgen dafür, daß zuerst die beiden nachfolgenden `\fi`-Tokens expandiert werden. Dadurch wird die momentane *if*-Abfrage und die durch `\iftrue` begonnene Abfrage beendet, bevor TEX das `\iffalse` expandiert.

```
11 \iffalse ja\else nein\fi
12 nein\fi
13 nein
```

Im Endeffekt sorgt dann dieses `\iffalse` dafür, daß der vom Benutzer des Makros `\ifKGR` angegebene *else*-Zweig abgearbeitet wird.

Auch für den positiven Fall, also für die Tokenfolge „KGR“, werden die Expansionsschritten gezeigt:

```
1 \ifKGR{KGR}ja\else nein\fi
2 \iftrue \if R\if G\if KKGR\fi\fi \else ... \fi ja\else nein\fi
3 \if R\if G\if KKGR\fi\fi \else ... \fi ja\else nein\fi
```

Diesmal findet das innerste `\if` mit „KK“ zwei Tokens vor, der Vergleich fällt positiv aus und diese beiden Tokens werden entfernt. Der Rest des *then*-Zweiges enthält jetzt noch die beiden verbliebenen Tokens „GR“ des Arguments. Durch die beiden verbliebenen *if*-Abfragen werden auch diese nacheinander verglichen und dabei durch den Vergleich entfernt.

```
4 \if R\if GGR\fi\fi \else ... \fi ja\else nein\fi
5 \if RR\fi\fi \else ... \fi ja\else nein\fi
6 \fi\fi \else ... \fi ja\else nein\fi
```

Übrig bleiben jetzt noch die `\fi`. Die drei `\if`-Vergleiche sind jeweils positiv ausgefallen, so daß TEX jetzt alle Tokens abarbeitet, bis er drei `\fi`-Tokens gelesen hat. Dabei werden alle *else*-Zweige ignoriert.

```
7 ja\else nein\fi
8 ja
```

Es verbleibt nur noch die äußerste Abfrage mit `\iftrue`, die immer wahr ist und deren *then*-Zweig jetzt abgearbeitet wird.

* * *

### Antworten auf die Fragen aus Teil II

In der letzten Folge habe ich in zwei Fußnoten Fragen gestellt, die ich natürlich nicht unbeantwortet lassen will:

1) Gibt es das Primitiv `\span` und wenn ja, wofür ist es gedacht? Ist dieses Primitiv expandierbar oder nicht expandierbar?

Natürlich gibt es ein Primitiv namens `\span`. Dieses Primitiv wird nur innerhalb der *Alignment*-Anweisungen `\halign` und `\valign` verwendet, um mehrere Tabellenspalten zu einer einzigen Spalte zu verschmelzen.[1] In dieser Verwendung ist `\span` *nicht* expandierbar.

In der Präambel einer Tabelle wird `\span` zur Expansion des nächsten Token zum Zeitpunkt der Präambel-Erstellung verwendet [1, S. 238]. Hier ist es in einem gewissen Sinne „expandierbar“, da `\span` danach nicht mehr in der erzeugten Präambel auftaucht.

```
\def\A{A} \count0=1
\halign{#\test\quad     \number\count0\quad\hfil
  &#\span\test\quad\span\number\count0\hfil\cr
  &\cr
\gdef\A{B}\global\count0=2\relax
  &\gdef\A{C}\global\count0=3\relax\cr
  &\cr}
\bye  % plain-TeX, nicht LaTeX
```

2) Welches Ergebnis erzeugt die Eingabe „`\quad Pluspunkt:`“, wenn das Makro `\quad` durch „`\def\quad{\hskip1em}`“ definiert wurde?

Das Ergebnis ist ... folgender Fehler:

```
*\quad Pluspunkt:
! Missing number, treated as zero.
<to be read again>
                   p
<*> \quad Plusp
               unkt:
```

Wie aus dem gezeigten Kontext und der Meldung, daß das `p` nochmals gelesen werden soll, ersichtlich ist, hat TEX von „Pluspunkt“ den Teil „Plus“ gelesen. Das Primitiv `\hskip` hat als Argument eine Glue-Spezifikation, also eine notwendige Längenangabe und zwei optional durch die Schlüsselwörter `plus` bzw. `minus` anzugebende Längen, die die zusätzliche Dehn- und Stauchbarkeit dieses „Glue“ angeben.

[1] In LATEX wird `\halign` zur Implementierung der `tabular`-Umgebung, `\span` zur Implementierung der Anweisung `\multicolumn` verwendet.





TEXs Scanner, der diese Angaben liest, ist „gierig“, d. h. er liest solange Tokens, bis er sicher ist, daß das nächste Token kein Bestandteil einer Angabe sein kann. Unsere neue Definition für `\quad` enthält nur die notwendige Länge. Außerdem wird am Ende der Makrodefinition nicht durch ein nicht expandierbares Token verhindert, daß der Scanner nach den Dehn- und Stauchbarkeitsangaben sucht. Beginnt der nach `\quad` folgende Text nun mit einem der beiden Schlüsselwörter, verwendet der Scanner einen Teil des Textes als Bestandteil der Glue-Spezifikation. Da jedoch nach dem „Plus“ keine Längenangabe folgt, also keine Zahl mit einer Einheit, wird ebendies als Fehler gemeldet.

Vermeiden kann man diese Art von Fehler, indem man in der neuen Definition des `\quad` am Ende ein `\relax` einfügt oder ein Glue-Register verwendet. Am besten ist es jedoch, die ursprüngliche, wohldurchdachte Definition des `\quad` unverändert zu lassen.

```
\def\quad{\hskip 1em\relax}
\newskip\quadskip  \def\quad{\quadskip=1em \hskip\quadskip}
```

Dieses gierige Scannen findet auch noch bei anderen Angaben statt. So sollte man bei den Glue-Spezifikationen auch auf die *fil*-Angaben und neben diesen auch bei `\hrule` und `\hbox` auf die möglichen Schlüsselwörter achten, um Fehler zu vermeiden. Besonders bei `fil`, `fill` und `filll` kann es unerwartet zu Fehlern kommen, da dies nicht drei Schlüsselwörter sind, sondern diese Wörter aus den beiden Worten `fil` und `l` zusammengesetzt werden. Da man Schlüsselworte beliebig groß und klein schreiben darf und zwischen zwei Worten optional Leerzeichen stehen dürfen, ist auch folgende Angabe erlaubt:

```
\hskip 0pT PlUs 1Fil L l   % = \hskip 0pt plus 1filll
```

Sogar erfahrene TEX-Benutzer können dabei noch so manchen Fehler begehen, der oft zu sehr unerwarteten Ergebnissen führen kann. Lieber Leser, versuchen Sie doch bitte, *vor* dem Ausprobieren in TEX, das Ergebnis der beiden folgenden Zeilen zu erraten.

```
\def\hfil{\hskip 0pt plus 1fil }
\hbox to\hsize{\hfil Leerzeichen, zentriert?\hfil}
```

Auch in diesem Fall hilft ein am Ende eingefügtes `\relax` dem Scanner auf die Sprünge und hält ihn davon ab, den Text unerlaubt „anzuknabbern“. Da der Scanner alle nachfolgenden `l` liest, also nach den maximal erlaubten zwei

zusätzlichen `l` für `filll` nicht einfach aufhört, kann man hiermit ebenso unerwartete Ergebnisse erzielen, diesmal in Form einer netten Fehlermeldung[2]:

```
\hskip 0pt plus 1fil l l l
```

Auch bei Linienangaben kann man die Schlüsselwörter in beliebiger Reihenfolge und beliebig oft angeben. Als kleine Übung kann man auch vor dem Ausprobieren erraten, wie breit und dick folgende Linie gezeichnet wird.

```
\hrule height 10pt depth 10pt width 0pt depth 0pt
       width 2pt height 1pt
```

Auch bei Makros, die eine `\hrule` oder `\vrule` am Ende haben, besteht die Gefahr, daß Teile des nachfolgenden Textes unerwartet als Linien-Spezifikation verwendet werd.en Hier kann man auch wieder mit einem `\relax` oder einem anderen nicht expandierbaren Token Abhilfe schaffen.

* * *

Dies wäre es für Teil III. Um die Wartezeit zum nächsten Teil dieser Serie zu verkürzen und gleich auf ihn einzustimmen, folgt hier ein kleines Rätsel: Was erzeugen die beiden folgenden Zeilen?

```
\def\1#1\fi#2#3{\fi\ifx.#3#1#2\else#2\1#3#1\fi}
\if00\1r\fi Jeukscta~ha~n{\TeX}o~trhe!.
```

### Literatur

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

[2] Donald E. Knuth: *TEX—The Program*; Addison-Wesley Publ., Reading, Mass., 1991.

[2] Tip: Nach dieser Fehlermeldung die zusätzliche Fehlererklärung durch Eingabe eines `h` abfragen!





## Das CJK-Paket für LaTeX 2ε

Werner Lemberg

Mit MULE (multilingual Emacs) kann man Texte in mehreren Sprachen schreiben. Es ist im besonderen entwickelt worden, um asiatische Sprachen in verschiedenen Kodierschemata zu editieren. So ist es zum Beispiel möglich, Chinesisch in der vereinfachten Schrift (jiǎntǐzì 简体字 in GB-Kodierung) und in der traditionellen Form (fántǐzì 繁體字 in Big-5-Kodierung) gleichzeitig darzustellen, obwohl sich die Kodierungsbereiche überlappen.
Das CJK-Paket (Chinesisch/Japanisch/Koreanisch) ist das Pendant für LATEX 2ε. Die meisten CJK-Kodierungen sind implementiert, das interne Kodierschema von MULE wird durch einen kleinen Präprozessor unterstützt. Außerdem enthält es Module für Unicode und CEF (Chinese Encoding Framework), das im Prinzip eine Eingabemethode für die CNS-Kodierung (Chinese National Standard, zirka 48 000 chinesische Schriftzeichen) ist.
CJK ist natürlich nicht auf MULE beschränkt. Jeder Editor, der eine der CJK-Sprachen unterstützt, kann verwendet werden.
Im CJK-Paket sind Hilfsprogramme enthalten, die chinesische TrueType-Fonts und CJK-Bitmap-Zeichensätze in `.pk`-Dateien konvertieren. Zusätzlich werden Sprachmodule für CJK-Sprachen zur Verfügung gestellt, die in Zusammenarbeit mit dem *Koma-Script*-Paket syntaktisch korrekte Überschriften produzieren.
Dieser Artikel beschreibt Version 3.0.1 des Pakets.

### Einführung

TEX erlaubt seit Version 3.x bekanntlich 8-Bit-Eingabe und -Zeichensätze. Jedoch ist das für die CJK-Sprachen (Chinesisch/Japanisch/Koreanisch) immer noch zuwenig, da bei diesen die Anzahl der Schriftzeichen 256 deutlich übersteigt. Eine generelle Erweiterung von TEX mittels Unicode [1], genannt „Projekt Omega“[1], befindet sich im Entwicklungsstadium. Unicode ist aber für viele alte chinesische Zen-Texte [2], um ein Beispiel zu nennen, nicht ausreichend. Des weiteren ist die Zusammenlegung von identischen beziehungsweise fast identischen Schriftzeichen aus verschiedenen Sprachen[2] nicht unproblematisch für linguistische Texte, die jede Variante von Schriftzeichen erfassen müssen. MULE, die multilinguale Erweiterung des mächtigen Editors Emacs, verwendet daher intern ein anderes Kodierungsprinzip, auf das später im Detail eingegangen wird.

[1] siehe `ftp://ftp.ens.fr/pub/tex/yannis/omega`
[2] Unicode hat rund 23 000 vereinheitlichte CJK-Schriftzeichen.

Das CJK-Paket ist ein Versuch, LaTeX 2ε-Unterstützung für Multibyte-Kodierungen zu Verfügung zu stellen. Es wurde von Anfang an universell ausgelegt, um neue Kodierschemata leicht hinzufügen zu können (was auch im Laufe der Entwicklung geschehen ist). Unter anderem gibt es Module für GB und Big-5 (Chinesisch), JIS und SJIS (Japanisch) sowie KS (Koreanisch). In einem der nächsten Abschnitte wird ausführlicher auf das *Chinese Encoding Framework (CEF)* von Christian Wittern sowie auf das Unicode-Modul eingegangen.

Meines Wissens gibt es nur einen einzigen frei verfügbaren Editor, der verschiedene Kodierschemata gleichzeitig darstellen kann, nämlich MULE. Aus technischen Gründen können TeX und LaTeX derart erzeugte multilinguale Dateien nicht direkt verwenden, sondern müssen sie mit einem Präprozessor aufbereiten (näheres weiter unten).

Abgerundet wird das Paket durch eine Reihe von Hilfsprogrammen, um CJK-Zeichensätze verschiedenster Herkunft in für TeX brauchbare Formate umzuwandeln. Ganz allgemein können Bitmap-Fonts im HBF-Format [3] sowie True-Type-Fonts, jedoch hierbei nicht alle Formate, verwendet werden. Fast alle frei verfügbaren Fonts findet man auf dem Software-Server `ftp.ifcss.org`.

Die jeweils aktuellste Version des CJK-Pakets ist auf den CTAN-Servern im Unterverzeichnis `tex-archive/language/chinese/CJK` zu finden. Dazu passende Font-Pakete finden sich in `tex-archive/fonts/CJK`.

### Asiatische Kodierschemata

Asiatische Schriftzeichen können nicht komplett mit einem Byte pro Zeichen dargestellt werden. Mindestens zwei Bytes sind notwendig, und die meisten gängigen Kodierschemata (GB, Big-5, JIS, KS, etc.) verwenden einen bestimmten Bereich für das erste Byte (normalerweise `0xA1` bis `0xFE` oder einen Teil davon), um zu signalisieren, daß dieses und das nächste Byte ein asiatisches Schriftzeichen darstellen. Das bedeutet im weiteren, daß gewöhnlicher ASCII-Text mit Zeichen zwischen `0x00` und `0x7F` davon unberührt bleibt und die meisten Zeichen des erweiterten ASCII-Zeichensatzes für eine CJK-Kodierung verwendet werden können.

Tabelle 1 zeigt alle implementierten Kodierschemata. Ein paar Besonderheiten zu dieser Tabelle:

- Manche Kodierschemata (Big-5, SJIS) haben „Löcher“ im Bereich des zweiten Bytes (Big-5 belegt beispielweise nur die Bereiche `0x40` bis `0x7E` und `0xA1` bis `0xFE`).





| Kodierung | Erstes Byte | Zweites Byte | Drittes Byte |
|---|---|---|---|
| GB | `0xA1`–`0xF7` | `0xA1`–`0xFE` | — |
| Big-5 | `0xA1`–`0xF9` | `0x40`–`0xFE` | — |
| JIS | `0xA1`–`0xF4` | `0xA1`–`0xFE` | — |
| SJIS | `0xA1`–`0xFE` | `0x40`–`0xFC` | — |
| KS | `0xA1`–`0xFD` | `0xA1`–`0xFE` | — |
| UTF-8 | `0xC0`–`0xEF` | `0x80`–`0xBF` | `0x80`–`0xBF` |
| CNS | `0xA1`–`0xFE` | `0xA1`–`0xFE` | — |

Tabelle 1: Kodierschemata

- SJIS, auch MS-Kanji genannt, besteht eigentlich aus zwei ineinander verschachtelten Kodierungen: den sogenannten Halfwidth-Katakana (Kodierschema JIS X0201-1976, 1-Byte-Sequenzen im Bereich `0xA1` bis `0xDF`) und dem auf andere Code-Punkte gelegten 2-Byte-Standard-Schema JIS X0208-1990.
- UTF 8 (Unicode Transformation Format 8), auch UTF 2 oder FSS-UTF genannt, ist eine spezielle Darstellung von Unicode beziehungsweise ISO 10646. Es werden ebenfalls Multibyte-Sequenzen unterschiedlicher Länge verwendet, jedoch sind im CJK-Paket nur 2-Byte- und 3-Byte-Sequenzen (welche Unicode komplett abdecken) implementiert. ASCII-Zeichen werden ohne Änderung als 1-Byte-Sequenzen übernommen. Diese Tatsache ermöglicht erst die Verwendung mit TeX.
- Per definitionem hat CNS 16 Ebenen mit jeweils 94×94 Zeichen. Derzeit sind sieben Ebenen belegt (CNS1 bis CNS7), eine achte Ebene soll in Entwicklung sein.
- Big-5 und SJIS können nur mit Schwierigkeiten direkt in TeX eingegeben werden, da die zweiten Bytes der Kodierschemata für TeX kritische ASCII-Zeichen enthalten, nämlich `{`, `}` und `\`. Die Klammern umschließen bekanntlich einen Block, der Backslash leitet einen TeX-Befehl oder ein TeX-Makro ein. Im CJK-Paket sind jedoch Präprozessoren enthalten (`Bg5conv`, `SJISconv` und `mule2cjk`), welche dieses Problem praktisch unsichtbar für den Anwender machen. Auf `mule2cjk` wird später noch genauer eingegangen.

Für weitere Details verweise ich auf [4], wo alle in Verwendung stehenden beziehungsweise in Verwendung gewesenen Kodierschemata ausführlich dargestellt sind.

### Das interne Kodierschema von MULE

Der Trick der internen Kodierung von MULE besteht einfach darin, jedes Nicht-ASCII-Zeichen mit einem Präfix-Byte im Bereich `0x80` bis `0x9F` zu versehen[3] und die jeweilige, dem Zeichen eigene Kodierung unverändert zu übernehmen. Ausnahme davon sind Big-5 kodierte Zeichen, welche in zwei Pseudo-Kodierungen zerlegt werden, damit die zweiten Bytes im Bereich `0xA1` bis `0xFE` liegen. Auf diese Weise können bis zu 32 unterschiedliche Kodierschemata dargestellt werden, ohne miteinander in Konflikt zu geraten. SJIS wird nicht in Kombination mit anderen Kodierschemata unterstützt, da dieses zu JIS äquivalent ist.

Um nun diese Kodierungen in TeX darstellen zu können, konvertiert das Programm `mule2cjk` die Präfix-Bytes für CJK-Kodierungen in Makros der Gestalt `\CJKenc{...}`, die in den laufenden Text[4] eingefügt werden. Die anschließenden zwei Bytes werden einfach übernommen, das erste Byte direkt als erweitertes ASCII-Zeichen in TeXs `^^xy`-Notation, das zweite als Dezimalzahl gefolgt von einem Punkt.

Beispiel:

„Grüß Gott“ in traditioneller chinesischer Schrift (Big-5-Kodierung):

„你好“ (Nǐhǎo, wörtlich „Du gut“)

wird nach der Verarbeitung durch `mule2cjk` dargestellt als

```
^^80Bg5.^^a765.^^a6110.
```

Das aktive Zeichen `^^80` ist definiert als

```
\def^^80#1.{\CJKenc{#1}}
```

Es wählt also die Big-5-Kodierung aus. Die anderen aktiven Zeichen zusammen mit ihren Argumenten wählen dann den jeweiligen Subfont und daraus das entsprechende Schriftzeichen.

Etwas anders verhält es sich mit den verschiedenen 1-Byte-Kodierungen wie Latin-1. Diese werden, völlig unabhängig von den CJK-Kodierungen, direkt in LaTeX 2ε-Makros übersetzt. Mit anderen Worten, `mule2cjk` kann auch für Texte verwendet werden, die z. B. nur Latin-2-Zeichen verwenden, ohne daß

[3] In Wirklichkeit sind auch zwei Präfix-Bytes für private, also benutzerdefinierte Kodierungen möglich.

[4] In Gestalt von aktiven Zeichen. Alle ersten Bytes der verschiedenen CJK-Kodierungen sind aktiv.





das CJK-Paket selbst geladen werden müßte. Auch hier werden aktive Zeichen verwendet, um die Funktionalität in `verbatim`-Umgebungen sicherzustellen.

### Das CJK-Paket im Detail

Zur Erläuterung hier ein kleines Beispiel, wie `CJK.sty` in einem LaTeX 2ε-Dokument verwendet werden kann:

```
\documentclass{article}
\usepackage{CJK}
\usepackage{pinyin}
\begin{document}
\begin{CJK}{Bg5}{fangsong}

我很喜歡吃中國飯。

\Wo3 \hen3 \xi3\huan1 \chi1 \Zhong1\guo2\fan4.

Ich esse gerne Chinesisch.

\end{CJK}
\end{document}
```

Ausgedruckt erscheint es etwa so:

我很喜歡吃中國飯。
Wǒ hěn xǐhuān chī Zhōngguófàn.
Ich esse gerne Chinesisch.

Wie man aus diesem Beispiel ersehen kann, sind im Prinzip nur zwei Schritte erforderlich, um Chinesisch eingeben zu können: zuerst wird mittels

```
\usepackage{CJK}
```

das Paket-File `CJK.sty` geladen, das wiederum bei Bedarf eine Reihe anderer Dateien lädt. Dann wird in der Regel am Anfang des Dokuments eine CJK-Umgebung mit `\begin{CJK}{Bg5}{fangsong}` eingeleitet und am Ende mit `\end{CJK}` geschlossen. `Bg5` gibt die Kodierung an, `fangsong` den verwendeten Font. Falls das CJK-Paket zusammen mit MULE verwendet wird, ist der zweite und dritte Parameter leer, da `mule2cjk` stattdessen `\CJKenc`-Makros einfügt und die CJK-Fonts über den Befehl (`\CJKencshape`) definiert werden müssen, falls nicht Standardwerte verwendet werden sollen.

`\usepackage{pinyin}` lädt `pinyin.sty`, welches ebenfalls ein Bestandteil des CJK-Pakets ist. Es ermöglicht die Pīnyīn-Eingabe und somit auch Fälle wie lǚlǜ.

Zusätzlich zur `CJK`-Umgebung gibt es `CJK*`, welche Leerzeichen und Zeilenenden nach CJK-Zeichen unterdrückt, da diese in reinem chinesischen Fließtext nicht vorkommen.

*Zeilenumbruch*

`CJK.sty` verhindert, daß bestimmte CJK-Sonderzeichen nicht am Zeilenende oder Zeilenanfang stehen. Dazu gehören alle Arten von Klammern und viele Interpunktionszeichen. Im Japanischen gibt es außerdem eine Reihe von Hiragana und Katakana, die nicht am Zeilenende stehen dürfen (*kinsoku shori* 禁則処理). Auch das wird von `CJK.sty` berücksichtigt. Die Interpunktionszeichenbehandlung geschieht für den Benutzer unsichtbar, kann aber bei Bedarf kontrolliert werden. Technisch gesehen fügt `CJK.sty` normalerweise zwischen zwei CJK-Zeichen elastischen Zwischenraum (`\hskip`) ein, an dem ein Zeilenumbruch möglich ist, bei Sonderzeichen aber nicht. Dies wird durch die Emission von praktisch unsichtbaren `\kern`-Befehlen realisiert (`\kern 1sp` bzw. `\kern 2sp`), die dem nächsten Zeichenmakro den Typ des gerade verarbeiteten Zeichens signalisiert. Ob es ein Interpunktionszeichen ist, wird tabellarisch ermittelt.

*Unicode*

Unicode-kodierte Zeichen können, da 16-Bit breit, nicht direkt mit TeX verarbeitet werden, sondern müssen in UTF 8-transformierter Form eingegeben werden. Diese Multibyte-Kodierung hat den Vorteil, daß sie gegenüber Zeichen in ASCII völlig transparent ist und somit in die heutige 8-Bit-Welt bestens integriert werden kann. Für 2-Byte-Werte, die den Unicode-Bereich `0x80` bis `0x7FF` abdecken, ist das erste Byte aus dem Bereich `0xC0` bis `0xDF`; für 3-Byte-Werte, die den restlichen Unicode-Bereich (`0x800` bis `0xFFFF`) abdecken, ist das erste Byte aus dem Bereich `0xE0` bis `0xEF`. Die anderen Bytes liegen zwischen `0x80` und `0xBF`. Diese auf den ersten Blick seltsam anmutende Aufteilung hat unter anderem den Vorteil, daß man stets eindeutig den Anfang einer Multibyte-Sequenz finden kann, egal wo man in den Datenstrom einsteigt.

Das CJK-Paket unterstützt den gesamten Unicode-Bereich und nicht nur den Abschnitt mit CJK-Zeichen, sofern man die geeigneten Fonts zur Verfügung hat. In einer späteren Version ist geplant, virtuelle Fonts zu konstruieren, welche die cm-Fonts von TeX auf Unicode abbilden. Ein Nachteil dieser Methode ist das Fehlen von Kerning-Paaren, sofern sich die Zeichen in verschiedenen





Subfonts befinden. Nur ein auf 16-Bit-breiten Zeichen basierendes TeX, wie zum Beispiel Omega, wird in der Lage sein, korrektes Kerning für Unicode zu liefern.

*CNS*

Christian Wittern (`wittern@conline.central.de`), ein ehemaliger Mitarbeiter von IRIZ [2], hat das *Chinese Encoding Framework (CEF)* entwickelt. CEF ermöglicht die Einbindung von CNS-11643-1992-kodierten Zeichen in andere Kodierschemata mit Hilfe von SGML-Makros der Gestalt `&Cx-yyyy;`, die einfach in den Fließtext eingebettet werden. `x` stellt die CNS-Ebene dar und `yyyy` den Zeichenkode in hexadezimaler Graphic-Left-Darstellung.[5] Er schrieb außerdem *KanjiBase for Windows*, mit dessen Hilfe man CNS-kodierte Zeichen eingeben kann. Pīnyīn- und Four-Corner-Eingabemethoden sind bereits implementiert, weitere sind in Arbeit.

*Font Definition Files*

Für asiatische Schriften, die aus rund 30 bis 55 TeX-Subfonts bestehen, definiert `CJK.sty` eine Reihe von zusätzlichen Fontladefunktionen (z. B. `CJK` oder `CJK-fixed`), die den ursprünglichen sehr ähnlich sind, jedoch den jeweils benötigten Subfont auswählen. Ohne ins Detail gehen zu wollen, sei hier ein Beispiel für die Definition eines GB-kodierten Fonts im Fangsong-Stil gezeigt:

```
\DeclareFontFamily{U}{GB}{}
\DeclareFontShape{U}{GB}{m}{fangsong}{<-> CJK * gsfs14}{}
\DeclareFontShape{U}{GB}{bx}{fangsong}{<-> CJKsub * GB/m/fangsong}{}
```

Die Subfonts haben in diesem Beispiel die Namen `gsfs1401` bis `gsfs1432`.

*Zeichensätze*

TeX kann bekanntlich 256 Zeichensätze zu je 256 Zeichen laden. Das mag viel erscheinen, ist es aber nicht, da der gleiche Font in zwei Größen als zwei Zeichensätze gilt. Alle CJK-Zeichensätze enthalten jedoch mehr als 256 Zeichen, so daß ein kompletter Satz 30 bis 55 TeX-Fonts benötigt. In einer früheren Chinesisch-Implementation (*poor man's Chinese*, pmC) wurden alle TeX-Fonts eines CJK-Zeichensatzes auf einmal geladen. Waren dann verschiedene Größen gefragt, konnte das Limit an Fonts schnell erreicht werden.[6] LaTeX 2ε ermöglicht

[5] Wittern definiert noch weitere SGML-Makros, um auch Unicode und private Kodierungen eingeben zu können, siehe dazu [5].

[6] Verschärfend kam dazu, daß bei pmC in jedem TeX-Font einige Positionen frei geblieben sind, was natürlich die Anzahl der benötigten Fonts erhöht.

es nun, nur die tatsächlich benötigten Fonts zu laden, was eine deutliche Erleichterung bedeutet. Erkauft wird dieser Vorteil mit einer enormen Verlangsamung bis zum Faktor 50, denn es muß für fast jedes Zeichen der Font gewechselt werden, falls ein Text beispielsweise nur aus chinesischen Zeichen besteht.

Mit dem CJK-Paket werden drei Hilfsprogramme mitgeliefert, die asiatische Bitmap-Fonts und chinesische TrueType-Fonts in für TeX verständliche Formate konvertieren, nämlich in PostScript und in das `.pk`-Format. Mitgeliefert werden auch Skript- bzw. Batch-Dateien, die das Erzeugen von Fonts „on the fly“ ermöglichen, so daß nicht komplette Sätze von vielleicht nie gebrauchten Subfonts im voraus zu berechnen sind. Anzumerken ist, daß die Verwendung von PostScript-Fonts noch sehr speicherintensiv ist, da `dvips` derzeit nur ganze PostScript-Fonts zum Drucker schickt und nicht die benötigten Zeichen allein, wie es schon für `.pk`-Fonts gemacht wird. Dadurch können die Ausgabedateien schon auf mehrere Megabyte anwachsen.[7] Die nächste Version von `dvips` soll dieses Manko allerdings nicht mehr haben.

Auf den CTAN-Servern findet man fertig zusammengestellte Font-Pakete für CJK. Weitere CJK-Fonts findet man auf dem ftp-Server `ftp.ifcss.org` und dessen Spiegeln (z. B. `cnd.org` oder `kth.se`). Dort sind praktisch alle public domain vorhandenen chinesischen Fonts gesammelt. Auch ein paar koreanische und japanische Bitmap-Fonts sowie CJK-Unicode-Fonts liegen dort.

*Sprachanpassung*

Die Standardklassen von LaTeX 2ε sind auf den englischsprachigen Benutzer (und hier wiederum auf den amerikanischen) zugeschnitten. Jedoch wurden im Rahmen der Internationalisierung Möglichkeiten geschaffen, durch Neubelegung einiger weniger Makros die Sprache zu ändern. Für viele Sprachen gibt es landesspezifische Pakete, die der Benutzer zu Beginn des Dokuments einbindet[8] und die intern diese Makros modifizieren. Die große Ausnahme bildeten bis jetzt die meisten asiatischen Sprachen, die anderen Numerierungskonventionen folgen als Englisch und vor allem die Zwischenräume anders setzen. Selbst im Englischen können nicht alle Möglichkeiten verwendet werden. So wird zum Beispiel „Chapter 1“ als Beginn einer Kapitelüberschrift unterstützt, „First Chapter“ jedoch nicht. In Zusammenarbeit mit Markus

[7] Abhilfe schafft hier das Hilfsprogramm `fontload`, welches die PostScript-Ausgabedatei nach den verwendeten Zeichen der jeweiligen Fonts durchsucht und neue, temporäre Zeichensätze konstruiert, die eben nur die gefundenen Zeichen enthält. Dieses Verfahren ist jedoch sehr langsam.

[8] Beispielweise wurde in diesem Artikel `german.sty` verwendet.





Kohm (`Markus_Kohm@ka2.maus.de`) wurde das von ihm gewartete und weiterentwickelte *Koma-Script*-Paket so erweitert, daß chinesische Gliederungsüberschriften möglich sind:[9]

Schreibt man

```
\part{中國飯\ (Chinesisches Essen)}
```

so erhält man

第一部　中國飯 (Chinesisches Essen)

Aktiviert werden die CJK-Gliederungsüberschriften durch den Befehl `\CJKcaption{xxx}` innerhalb einer `CJK`- oder `CJK*`-Umgebung. `xxx` lädt das jeweilige Sprachmodul `xxx.cap`, das die oben beschriebenen Anpassungen vornimmt. Die Namen der Sprachmodule spiegeln in der Regel das Kodierschema wieder, und so heißt das chinesische Modul in Big-5-Kodierung `Bg5.cap`. Die technische Realisierung erfolgte durch Einführung einer dritten Ebene innerhalb der Namensmakros.

Ebene 1 sind die wohlbekannten Standardmakros `\figurename` etc. Hier ersetzen normalerweise die landesspezifischen Pakete wie `german.sty` die englischen Namen mit landessprachlichen Wörtern.

Ebene 2 ist die Modifikation der Abschnittszähler. Das Makro `\thesection` soll chinesische Zählzeichen an Stelle von arabischen Ziffern verwenden.

Ebene 3 schließlich erlaubt die völlige Kontrolle über die Anordnung von Zählern, Überschriften und Zwischenräumen. Sie besteht aus den Befehlen der Form `\...format`, zum Beispiel `\chapterformat`, `\sectionformat`. Diese werden direkt von `\chapter` etc. verwendet.

Hier ein Beispiel für einen chinesischen `\part`-Befehl:

```
\newcommand\CJKnumber[1]{\ifcase#1\or
                         一\or二\or三\or四\or五\or
                         六\or七\or八\or九\or十\fi}

\newcommand\prepartname{第}
\newcommand\postpartname{部}
\renewcommand\thepart
  {\prepartname\CJKnumber{\value{part}}\postpartname}
\renewcommand\partformat{\thepart}
```

[9] Im Prinzip war das auch bisher schon mit den Standardklassen möglich, jedoch mußte man sehr tief in die Struktur der Standardklassen eingreifen, um optisch ansprechende Ergebnisse zu erzielen.

### Schlußbemerkung

Das CJK-Paket reizt NFSS bis zum Äußersten aus. Überlegenswert wäre eine vereinfachte Schnittstelle, die speziell auf die Auswahl von Subfonts ausgelegt ist. Chen Hung-Yih hat so etwas ähnliches für plain-TeX implementiert [7]. Eine solche Arbeit wäre wahrscheinlich durch die Einführung von Omega Zeitverschwendung.

Für weitere Fragen und Anregungen stehe ich unter der auf Seite 61 angegeben Adresse gerne zur Verfügung.

### Literatur

[1] *The Unicode Standard Vol. 1 u. 2.*, Addison-Wesley Publ., Reading, MA, 1992.
*Die meisten aktualisierten Tabellen und Querreferenzen für* CJK *sind mittels ftp von dem Software-Server* `ftp.unicode.org` *erhältlich.*

[2] *ZenBase CD1*, International Research Institute for Zen Buddhism (IRIZ), Hanazono University (花園大学国際禅学研究所), Kyōtō, 1995.
*Die weltweit erste CD-ROM mit einer Sammlung chinesischer buddhistischer Texte. Außerdem enthält sie* KanjiBase for Windows*, das Eingabe-Tool für CEF, und viele andere Programme, die für ostasiatische Studien von Nutzen sein können. Siehe auch* `http://www.iijnet.or.jp/iriz/irizhtml/irizhome.htm`.

[3] *Hanzi Bitmap Font File Format Ver. 1.1 (1994). Als* ASCII*-Datei oder im HTML-Format erhältlich auf dem ftp-Server* `ftp.ifcss.org`*. Eine Implementation eines HBF-API (in C) befindet sich ebenfalls dort.*

[4] Ken Lunde, *CJK.INF – Online Companion to "Understanding Japanese Information Processing."*
*Zu finden in* `ftp://ftp.ora.com/pub/examples/nutshell/ujip/doc/cjk.inf`*. Eine sehr ausführliche Referenz aller CJK-Kodierungen von einem der Experten auf diesem Gebiet.*

[5] Christian Wittern, *The IRIZ KanjiBase*, in: *The electronic Bodhidharma* (電子達摩) Vol. 4 (June 1995), Kyōtō.
*Diese Ausgabe des IRIZ-Journals wird mit [2] ausgeliefert und ist gänzlich der wissenschaftlichen Edition alter asiatischer Texte gewidmet.*

[6] Urs Widmer, *Besonderheiten der chinesischen und japanischen Textgestaltung mit TeX*, in: *Offizin*, hrsg. von DANTE e.V., Addison-Wesley Verlag, Bonn, 1994.





[7] ChiTeX. *Zu finden auf dem ftp-Server* `dongbo.math.ncu.edu.tw` *im Verzeichnis* `/pub/yih/chitex`*. Dieses Makro-Paket ermöglicht Chinesisch (allerdings nur in Big-5-Kodierung) sowohl für plain-TeX als auch für LaTeX. Es ist deutlich schneller als das* `CJK.sty`*, da z. B. die Subfont-Auswahl fest kodiert ist. ChiTeX benutzt dieselben Fonts wie das CJK-Paket.*

**Was Sie schon immer über TEX wissen wollten . . .**

## Kapitälchen in Überschriften

Bernd Raichle

Von mehreren LaTeX-Benutzern wurde ich in der Vergangenheit oft danach gefragt, ob es keine fette Kapitälchen gibt. Bei der Verwendung von Eigennamen in Kapitälchen, wie

```
\newcommand{\Unix}{\textsc{Unix}}
```

die man in Überschriften benutzt, erhält man von LaTeX 2ε eine Warnung, daß einige der angeforderten Schriftformen nicht verfügbar sind und LaTeX 2ε daher auf andere ausweichen mußte. Da die Standard-CM-Schriften keine fetten Kapitälchen enthalten, ersetzt LaTeX 2ε die fetten Kapitälchen durch die fette Standardschrift. Somit erhält man für `\section{\Unix}` auf dem Papier leider nur „**Unix**“ statt „**UNIX**“ oder wenigstens einem nicht fett geschriebenem „UNIX“.

In vielen Fällen ist es besser, auf Fettschrift zu verzichten, dafür aber die Kapitälchen beizubehalten. In LaTeX 2ε ist dies durch die Zeilen

```
\DeclareFontShape{OT1}{cmr}{bx}{sc}{<->ssub*cmr/m/sc}{}
\DeclareFontShape{OT1}{cmr}{b}{sc}{<->ssub*cmr/m/sc}{}
\DeclareFontShape{T1}{cmr}{bx}{sc}{<->ssub*cmr/m/sc}{}
\DeclareFontShape{T1}{cmr}{b}{sc}{<->ssub*cmr/m/sc}{}
```

in der Dokumentpräambel sehr einfach möglich. Die DC-Schriften werden erst ab Version 1.3 standardmäßig fette Kapitälchen enthalten, so daß man im Moment noch die letzten beiden Zeilen benötigt. Durch die in der `\Declare-FontShape`-Deklaration verwendete Funktion `ssub` unterdrückt man die oben genannte Warnung von LaTeX 2ε, daß es Schriftformen durch andere ersetzen mußte. Wollen Sie diese Warnungen weiterhin erhalten, so ersetzen Sie bitte in den oben gezeigten Zeilen `ssub` durch die Funktion `sub`. Weitere Informationen zu den Möglichkeiten des *New Font Selection Scheme* finden Sie in der mit LaTeX 2ε mitgelieferten Dokumentation [1] undim Buch *Der LaTeX-Begleiter* [2, Kapitel 7.2].





**Literatur**

[1] LaTeX3 Project Team: *LaTeX 2ε font selection*; Dezember 1995. Als Datei `fntguide.tex` in der LaTeX 2ε-Verteilung enthalten.

[2] Michel Goossens, Frank Mittelbach und Alexander Samarin: *Der LaTeX-Begleiter*; Addison-Wesley Verlag, Bonn, 1994.

# TEX-Beiprogramm

## Das alternative (LA)TEX-Glossar

Andreas Dafferner, Luzia Dietsche, Bernd Raichle,
Volker RW Schaa, Rainer Schöpf

Vielen Anfängern und Umsteigern von anderen Textverarbeitungsprogrammen ist die große Anzahl von Befehlen in TEX und LATEX der Haupthinderungsgrund für eine schnelle und effiziente Einarbeitung. Das hier abgedruckte Glossar soll einen ersten einfachen Einstieg in die komplexe Befehlsstruktur auf eine leicht verständliche Weise ermöglichen. Bei Bedarf werden weitere Folgen auf Schwerverständliches oder Wissenswertes eingehen.

| | |
|---|---|
| `\addtolength` | Folge von `\stretch` |
| `\addvspace` | Dachgeschoßausbau |
| `\amalg` | Attribut zu `\fill` |
| `\bar` | Alternative zum Bezahlen mit Plastikgeld |
| `\begin{minipage}` | Anfänger im Hotelgewerbe |
| `\begin{picture}` | Aufforderung/Anweisung an Künstler |
| `\bibindent` | neue Zahnpastasorte |
| `\bigcup` | ein Becher Kaffee (*siehe auch* `\cup`) |
| `\bottomfraction` | Hinterwäldler im Bundestag (*siehe auch* `\topfraction`) |
| `\catcode` | etwas für's „Katzeklo“ |
| `\centerline` | Mittelstreifen |
| `\clearpage` | Aufforderung an einen Hoteldiener, den Weg freizumachen |
| `\clubpenalty` | Strafe, die Sie erhalten, wenn Ihr Lebensabschnittsgefährt [neutrale Form!] Sie mit einem Club (engl. Prügel, Schlagstock) trifft;<br>Strafe, nicht Mitglied bei DANTE e.V. zu sein |





| | |
|---|---|
| `\clubsuit` | notwendiges Kleidungsstück, um in einen englischen Club Eintritt zu erhalten |
| `\cot` | zusammen mit Flügeln Bestandteil jeden Autos |
| `\cup` | eine Tasse Kaffee (*siehe auch* `\bigcup`) |
| `\deadcycles` | kaputte Fahrräder |
| `\Delta` | US-amerikanische Fluggesellschaft |
| `\diamondsuit` | Kleidungsstück für die Diamantene Hochzeit |
| `\discretionary` | chambre séparée |
| `\displaypenalty` | Verzeichnis der erhaltenen Vorstrafen |
| `\documentclass` | Grundschule für Dokumente |
| `\driver` | Jagdgehilfe |
| `\dump` | siehe `\muglue`, beim Menschen |
| `\eject` | Hinauswurf, falls Sie ohne `\clubsuit` angetroffen werden |
| `\encl` | männliche Nachkommen |
| `fancyheadings Package` | Schachtel für extravagante Hüte |
| `fil` | Taylor |
| `fill` | Zahnfüllung (*siehe auch* `\amalg`) |
| `\flat` | Reifenpanne |
| `floats lost` | „Mami, mein Luftballon ist weg!“ |
| `floats too large for page` | Luftballon war zu groß für Hoteldiener |
| `\flushbottom` | Toilettenspülung |
| `\flushglue` | Reste, die nach einer Spülung in der Toilette verbleiben |
| `\frac` | ab hier wird's feierlich (*siehe auch* `\clubsuit`) |
| `\framebox` | Aufbewahrungsort für Bilderrahmen |
| `glo` | Schreibfehler |
| `\halign` | Aufstellen zum großen Zapfenstreich |
| `\headsep` | häufig genutztes Utensil während der französischen Revolution |

`I can't find file` Aufschrei in der Verwaltung

`\immediate\openin` sofortiger Einlaß beim Fensterln

`\immediate\openout` falsches Fenster (Elternschlafzimmer)

`\input` Aufforderung zum Essen

`italic correction` Rettungsmethode für den *Schiefen Turm von Pisa*

`\Join` Aufforderung zum Eintritt bei DANTE e.V.

`\kill` „Spiel mir das Lied vom Tod“

`\lambda` Sonde im Katalysator

`\lastpenalty` letzte Warnung

`\LaTeX` Spielzeug für Fetischisten, *seltener auch:* alternative Verhütungsmethode

`\leaders` intelligentes Aussitzen

`\linepenalty` Strafe für's auf dem Strich gehen

`\makeindex` Aufforderung an einen Autor, endlich ein Stichwortverzeichnis bereitzustellen

`\markright` Aufforderung zum Kenntlichmachen von Rechtsextremisten

`\mathindent` neue Zahnpastasorte

`\medskip` Sprungweite von Medizinern

`Missing $ inserted` kleine Spende

`\models` Claudia Schiffer, Naomi Campbell, Linda Evangelista

`muglue` Kuhscheiße (*siehe auch* `\muskip`)

`\multiput` Hühnerfarm (*siehe auch* `\put`)

`\muskip` Abstand zwischen zwei `muglues`

`\newpage` Wechsel des Hoteldieners

`\newenvironment` Ausweg aus der ökologischen Katastrophe

`\noalign` rührt Euch!

`\noextrasgerman` Deutsche erhalten keine Sonderrechte

`\nolinebreak` in Reih' und Glied





`\nopagenumbers` Hoteldiener ohne Rückennummer

`Not a letter` Beschwerde eines Postboten

`\nu` jetzt aber auf!

`\obeylines` Kleingedrucktes

`\outputpenalty` Strafe für die Verunreinigung von öffentlichem Grund

`Overfull \hbox` Hose zu eng; übervolle Hose

`\pagegoal` erwartetes Trinkgeld im Hotelgewerbe

`\pageshrink` Verhalten von Hotelangestellten, wenn `\pagegoal` nicht erreicht wird

`\phi` Vieh (*siehe auch* `\varphi`)

`poolsize` Größe des Swimmingpools

`\poptabs` wenn die Tabs wieder rauskommen (*siehe auch* `\pushtabs`)

`\psi` unerklärliche Phänomene

`\pushtabs` wenn Sie mehrere Coregas ins Glas werfen (*siehe auch* `\poptabs`)

`pushtabs and poptabs don't match` Kukis rein, Coregas raus

`\put` Henne (*siehe auch* `\multiput`)

`\radical` Verfassungsfeind

`\raisebox` Aufzucht von Schachteln

`\Re` Ausdruck beim Kartenspiel

`\relax` don't worry, be happy!; britisches Gasthaus, von Galliern gerne besucht

`\rho` ungekocht (*siehe auch* `\varrho`)

`\samepage` Bitte, vom gleichen Hoteldiener wie beim letzten Besuch bedient zu werden

`\shiftamount` Niederschlagsmenge

`\shipout` Stapellauf; `\dump` vor `\flushbottom`

`\sideways` Krebsgang

`\spadesuit` Anzug einer Figur von Dashiell Hammett

| | |
|---|---|
| `\stretch` | Übung beim Aerobic |
| `\suppressfloats` | nur für Taucher; keine Rettung Schiffbrüchiger |
| `\swabfamily` | Schwarzwaldfamilie |
| `Tab overflow` | zuviele Tabs im Glas oder Glas zu klein (*siehe auch* `\pushtabs`) |
| `\today` | Billigmarke bei REWE |
| `Too many unprocessed floats` | zu viele nicht fliegende Luftballons |
| `\topfraction` | Mehrheit im Bundestag (*siehe auch* `\bottomfraction`) |
| `\topmargin` | Spitzensteuersatz |
| `\totalheight` | Höhe inkl. Haare, Schuhe (mit Absätzen) |
| `\triangleleft` | Default (*siehe auch* `\triangleright`) |
| `\triangleright` | Anweisung an den Musiker, in welcher Hand das Instrument zu halten ist |
| `Underfull \vbox` | zuwenig Menschen in einer kleinen Berghütte |
| `\usebox` | Recycling von Schachteln |
| `\usepackage` | Verpackungsanweisung |
| `\usepackage[5cm]` | kleines Päckchen, Parfümverpackung |
| `\vadjust` | hochhackige Schuhe |
| `\valign` | artistisches Kunststück im Zirkus |
| `\varphi` | Steak (*siehe auch* `\phi`) |
| `\varrho` | gekocht (*siehe auch* `\rho`) |
| `\voidbox` | Sorry Pandora |
| `whatsit` | weeß nich;<br>Wie bitte? Setzen? |
| `\widehat` | Kopfbedeckung für Großkopferte |
| `\widowpenalty` | Strafe, wenn Sie auf eine Witwe treffen (kann negativ sein) |





# Rezensionen

## 4allTEX – eine Ready-to-Run-CD auf dem Prüfstand

Rainer Hülse

Vor gut einem Jahr hörte ich durch Zufall von einer CD unserer TEXnischen Kollegen aus den Niederlanden ($\mathcal{NTG}$), die damals bereits die zweite Auflage einer Ready-to-Run-CD für TEX herausgebracht hatten. Da die erste Auflage in einer verhältnismäßig geringen Stückzahl von 250 Exemplaren erschienen war, nahm man hier in Deutschland kaum Kenntnis davon. So war ich wohl einer der wenigen, der diese CD-ROM in Deutschland testen konnte.[1]

Ich war sehr froh über diese Möglichkeit, TEX mit allen Möglichkeiten komplett von CD-ROM laufen zu lassen, da ich damals noch der aussterbenden EDV-Rasse angehört hatte, die den beliebten Dreikampf auf dem PC – mehr Leistung, mehr RAM, größere Festplatte – nicht mitgemacht hatte. Ich arbeitete noch immer auf meinem Steinzeit-Computer, den ich mir sechs Jahre zuvor als Non-plus-Ultra gekauft hatte (286er mit Herculeskarte, 1 MB RAM, 42 MB Festplatte). Windows? Das machte keinen Spaß. DOS-Anwendungen? Sorry, VGA card needed! Aber ich hatte ja TEX! TEX lief doch ganz prima auf meiner alten Kiste. Aber halt, alles lief auch nicht, wie es laufen sollte. Ein Blick auf die Versionsnummer verriet mir, weshalb: PCTEX 2.93. Das Ersetzen durch emTEX mit der TEX-Version 3.1415 hatte ich bisher noch nicht gewagt. Never change a running system. Vor allem dann nicht, wenn man das alte System aus Festplattenplatzgründen vorher vollständig löschen müßte, um das neue System zu installieren. Doch dann kam die Lösung!

4allTEX: 600 MB TEX pur auf dem neuesten Stand. So konnte ich endlich Schritt für Schritt mein altes TEX ersetzen und dabei auch noch meinen knappen Festplattenplatz entlasten.

Für die erste Installation benötigte ich nur wenige Minuten und etwa 50 KB Festplattenplatz. Ich hatte die etwas gewöhnungsbedürftige Benutzeroberfläche 4TEX vor mir. Bei näherer Betrachtung stellte ich fest, daß diese Oberfläche nur

[1] Anmerkung der Redaktion: Die CD ist für Mitglieder von DANTE e.V. über den Verein erhältlich, siehe Bestelliste.

mit 4DOS – entweder als Primary oder als Secondary Shell – arbeitet. Da ich bereits früher schlechte Erfahrungen mit 4DOS gemacht hatte – 4DOS läuft bei mir sehr instabil und neigt zu spontanen Abstürzen – und ich sowieso gewohnt bin, TEX aus Batchdateien zu starten, untersuchte ich, ob es für die wichtigsten Programme und Treiber Batchdateien gibt. Doch leider Fehlanzeige. Nun gut, Batchdateien verbrauchen nicht allzuviel Festplattenplatz. Also nahm ich meine alten Batchdateien und versuchte, sie an die Pfade der CD anzupassen. Doch da kam die nächste Überraschung: im Gegensatz zu PCTEX benutzt emTEX eine ganze Reihe von Umgebungsvariablen. Und diese möglichst vollständig aus den Dokumentationen, die auf der CD zum Glück reichlich vorhanden sind, herauszulesen, hat mich dann eine ganze Weile beschäftigt.

Leider hat man auch bei der dritten Auflage die 4TEX-Oberfläche unter 4DOS beibehalten. Das halte ich für den gravierendsten Mangel, auch wenn die Autoren dies als Vorteil darstellen. Denn auch auf meinem neuen Pentium-Rechner verhält sich 4DOS schlecht! Und das Problem habe wahrscheinlich nicht nur ich.

Nun war ich also soweit, die ersten Musterdateien zu TEXen. Zum Glück hatte ich noch mein altes System auf der Platte. Denn mit der neuesten Version der Preview- und Druckertreiber hat sich auch die Responsefile-Sprache geändert (sehr gut: die verkürzte Schreibweise für Pfade). Für die Anpassung würde ich sicher wieder einige Zeit an DOC-File-Studium brauchen. Aber ich wollte doch ein erstes Erfolgserlebnis. So druckte ich meinen Text mit einem älteren Treiber aus – und erlebte die nächste Überraschung: der Ausdruck sah völlig anders aus als frühere Ausdrucke desselben Textes. Es tauchten übergroße Leerzeichen im Text auf, und, was noch schlimmer ist, die Trennungen waren eine Katastrophe (Beispiel: erzeug-t). Wie kam das zustande?

Wieder folgten Stunden der Analyse. Schließlich sollten doch auch auf einer niederländischsprachigen CD, auf der Babel läuft, mit dem `german`-Paket auch die deutschen Trennmuster zur Verfügung stehen. Fehlanzeige! Es stehen zwar erfreulich viele Format-Files zur Verfügung, doch hat man zu ihrer Erzeugung offenbar nur niederländische und englische Trennmuster eingebunden, in der `language.dat` ist die Zeile `german dehyphen.tex` auskommentiert. (Die richtige Datei wäre sowieso `ghyph31.tex` gewesen!) Die Folge: ich mußte mir die wichtigsten Formate selbst erzeugen. Meine arme Festplatte! Außerdem wird die Formaterzeugung nicht besonders gut unterstützt. Die nötigen Files mußte ich mir aus verschiedenen Verzeichnissen selbst zusammensuchen, da die Dateien zur Formaterzeugung, wie ich sie auf der CD vorfand, nicht zueinander paßten.





Negativ machten sich auch die langen Suchwege auf der CD bemerkbar. Das fällt aber bei einem Pentium und einem Quad-Speed-Laufwerk nicht mehr ins Gewicht. Daß nicht alle Stylefiles aus dem CTAN-Archiv vorhanden sind, ist auch schade. Andere Sachen sind dafür doppelt oder dreifach auf der CD.

Die Standard-Fonts waren für viele Druckerauflösungen in Fontlib-Dateien vorhanden. Davon ist man bei der 3. Auflage wieder abgegangen. Jetzt liegen wieder normale Font-Verzeichnisse vor. Doch leider, oh Schreck, kann man sich nicht mehr darauf verlassen, daß sie noch komplett sind. So wunderte ich mich, daß für den HP Deskjet ein cm-Font (`cmssq8.pk`) fehlt. Aber da ja auch die Fonts für den Laserjet vorhanden sind, mußte ich nur die Umgebungsvariable auf diesen Pfad erweitern, und alles lief wie zuvor.

Zusätzlich existieren auch einige PostScript-Fonts im pk-Format, zum Beispiel Times. Leider funktionierte der Zugriff über virtuelle Fonts nicht, wie ich gehofft hatte, so daß ich mich auch in die Erstellung von virtuellen Fonts einlesen mußte, um diese selbst zu generieren. Auch normale PostScript-Fonts liegen in großer Anzahl vor. Leider konnte ich diese in der dritten Auflage noch nicht testen. Ich hoffe, daß der Zugriff auf die PostScript-Fonts nun besser geworden ist.

LATEX $2_\varepsilon$ läuft nur mit BigTEX, was mit einem 286er mit 1 MB RAM und dem nicht gerade kleinen CD-Treiber keinen Spaß gemacht hat, da jede Seite zirka eine Stunde zur Berechnung braucht. Aber das ist kein spezielles Problem der CD. Immerhin war es schon ein Vorteil, daß ich überhaupt LATEX $2_\varepsilon$ benutzen konnte. Ähnliche Probleme ergaben sich auch für Ghostscript, welches bei geladenem CD-Treiber überhaupt nicht ordentlich gearbeitet hat. So war ich gezwungen, Ghostscript auf der Festplatte zu installieren, um zumindestens Schriften drehen zu können. Bei komplizierteren PostScript-Files reicht aber der Speicher auch hier nicht aus.

Positiv ist noch zu vermerken, daß viele Utilities auf der CD vorhanden sind, public domain und shareware, die das Arbeiten mit TEX erleichtern. So sind METAFONT und BIBTEX ebenso vorhanden wie BM2FONT und *MakeIndex*. Als Rechtschreibprüfung ist AMSPELL recht nützlich.

Selbstverständlich kann ich hier nicht alle Features von 4allTEX besprechen, da die CD 1300 MB Software enthält. Verwiesen sei hier aber auf ein 162-seitiges Handbuch, das den CDs beiliegt. Die DOC-Files liegen bei der dritten Auflage auf einer zweiten CD, so daß die erste CD noch immer als Ready-to-Run-CD funktioniert, der Platz für die Dokumentationen aber zugunsten von Anwendungen frei wurde. Die DOC-Files auf der zweiten CD lassen sicher keine

Fragen offen. Und wenn es doch noch Fragen gibt, werden sie bestimmt in einer der Kommunikationslisten (`texhax`, `tex-nl`, `tex-mag` und `uktex`) beantwortet.

*Fazit*

Um nicht mißverstanden zu werden: Ich halte 4allTEX für eine sehr gute Idee. Die Kritik, insbesondere die Problematik mit 4DOS, soll von den Autoren, falls sie diese zu lesen bekommen, als konstruktive Kritik für hoffentlich noch viele Auflagen dienen.

Nach meinen Anpassungen läuft 4allTEX jetzt reibungslos und ich bin damit vollauf zufrieden. An Festplattenplatz benötige ich jetzt vier MB gegenüber 15 MB mit dem alten TEX. Das ist leider mehr, als ich erhofft hatte, allerdings ist die Einsparung von etwa 11 MB auch ganz schön. Auf meinem neuen Pentium habe ich zwar weniger Probleme mit der Hardware, aber meine Gigabyte-Platte ist nach wenigen Wochen auch schon wieder voll, so daß ich nach wie vor TEX von der CD laufen lasse.

An Arbeitszeit hatte ich bei der zweiten Auflage ungefähr zwei Monate gebraucht, um alle Batch- und Response-Dateien, die ich benötige, zu schreiben. Für die Anpassung an die dritte Auflage reichten dann ein paar Stunden. Ich mußte ein paar Formate für die emTEX-Version 4a (TEX-Version 3.14159) neu generieren und ein paar Pfade für die Fonts neu setzen. Auch der emTEX-Previewer und -Druckertreiber liegen in einer neueren Version vor, in der sich ein paar Parameter geändert haben. Aber das ist nicht so schlimm, denn für die Schnell-Ansicht und den schnellen Ausdruck benutze ich jetzt die Windows-Programme von der CD-ROM, die auch unter Windows95 gut funktionieren, und ansonsten greife ich halt noch eine Weile auf die alten Programme zurück.

Sicher gibt es immer noch einiges, was ich noch verbessern kann, schließlich bekommt man zigtausend Dateien auf der CD nicht so schnell in den Griff. Aber, wie gesagt, die wichtigsten Anpassungen sind fertig. Und ich habe eine Menge über TEX gelernt!





| *Vorteile* | *Nachteile* |
|---|---|
| ⊕ 1300 MB TEX pur | ⊖ vieles doppelt und dreifach auf der CD |
| ⊕ Ready-to-Run | ⊖ nur Formate mit niederländischen Trennmustern |
| ⊕ LaTeX 2ε | ⊖ es fehlen Stylefiles vom CTAN-Archiv |
| ⊕ viele Utilities (Ghostscript, Graphic Workshop, BM2FONT, TEXCAD, *MakeIndex*, AMSPELL u. v. m.) | ⊖ keine normalen Batch-Files, da nur für 4DOS |
| ⊕ viele Formate | ⊖ viele Fonts nur als MF-Files |
| ⊕ viele Fonts (pk, fli, ps) | ⊖ schlechte virtuelle Fonts für die Benutzung von PostScript-Fonts |
| ⊕ neueste Treiber | ⊖ schlechte Unterstützung bei Formatgenerierung |
| ⊕ METAFONT | ⊖ `exe`-Files in zu vielen Verzeichnissen verteilt (Probleme mit `path`-Länge!) |
| ⊕ BibTEX | ⊖ LaTeX 2ε nur mit BigTEX (sehr langsam) |
| ⊕ mehrere für TEX besonders geeignete Editoren | ⊖ CD-Treiber läßt nur wenig Hauptspeicher (zu wenig z. B. für Ghostscript) |
| ⊕ ausführliches Handbuch | |
| ⊕ viele DOC-Files auf der zweiten CD, z. T. auch als DVI-Files | |
| ⊕ minimale Hardware-Voraussetzungen | |

Tabelle 1: Vor- und Nachteile der 4allTEX-CD

# Spielplan

## Termine

**2.5.–4.5.1996** BachoTEX'96 „The World around TEX" – GUST Annual Meeting in Bachotek
Bachotek (Brodnica Lake District), Polen
Kontakt: Jolanta Szelatynska

**12.5.–16.5.1996** SGML Europe '96
Park Hotel, München
Kontakt:Tel.: +1/703/519-8160
Fax: +1/703/548-2867
munich96@aol.com

**24.5.–26.5.1996** NTUG Meeting
1996 Nordic TEX Users Meeting
Tallin, Estonia
Kontakt:Enn Saar
saar@aaai.ee

**25.6.–29.6.1996** 1996 Joint International Conference ALLC/ACH'99 (Association for Literary and Linguistic Computing, Association for Computers and the Humanities)
University of Bergen, Norwegen
Kontakt: Espen S. Ore

**28.7.–1.8.1996** TUG'96
Dubna (ca. 100 km nördlich von Moskau), Rußland
Kontakt: Irina Makhovaya

**24.9.–26.9.1996** EP96 – International Conference on Electronic Documents, Document Manipulation and Document Dissemination
Xerox Research Center, Palo Alto, Kalifornien, USA
Kontakt: Xerox Corporation





# Stammtische

*In verschiedenen Städten im Einzugsbereich von DANTE e.V. finden regelmäßig Treffen von TEX-Anwendern statt, die für Jeden offen sind. Wer gerne auch einen solchen Termin anbieten möchte, um sich mit anderen TEXies auszutauschen, schickt einfach die Adresse der Ansprechperson, die Adresse des Treffpunktes und den Zeitpunkt des Treffens zur Veröffentlichung an die Redaktion.*

**10587** Berlin
Rolf Niepraschk
Physikalisch-Technische Bundesanstalt
Abbestr. 2–12
Tel.: 030/34 81 316
niepraschk@ptb.de
*Gaststätte „Bärenschenke"*
*Friedrichstr. 124*
*Letzter Donnerstag im Monat, 19.00 Uhr*

**22527** Hamburg
Volker Huettenrauch
volker_huettenrauch@hh.maus.de
*HOPI, Oberstr. 3*
*Letzter Mittwoch im Monat, 18.00 Uhr*

**28359** Bremen
Martin Schröder
Tel.: 0421/62 88 13
ms@dream.hb.north.de
*Universität Bremen, MZH 4. Stock gegenüber den Fahrstühlen*
*Erster Donnerstag im Monat, 18.30 Uhr*

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

**42283** Wuppertal
Andreas Schrell
Erlenstr. 1
Tel.: 0202/50 23 54
Andreas_Schrell@rs.maus.de
*Gasthaus „Yol", Ernststr. 45*
*Zweiter Donnerstag im Monat, 19.30 Uhr*

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

**50931** Köln
Uwe Münch
Schmittgasse 92
51143 Köln
Tel.: 02203/8 71 11
muench@ph-cip.uni-koeln.de
*Zentrum für Paralleles Rechnen, Weyertal 80*
*Vierter Dienstag im Monat, 20.00 Uhr*

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

**69008** Heidelberg
Luzia Dietsche
Tel.: 06221/2 97 66
dante@dante.de
*China-Restaurant Palast*
*Lessingstr. 36*
*Letzter Mittwoch im Monat, 20.00 Uhr*

**76128** Karlsruhe
Klaus Braune
Tel.: 0721/6 08 40 31
braune@rz.uni-karlsruhe.de
*Universität Karlsruhe, Rechenzentrum*
*3. OG Raum 316*
*Zirkel 2*
*Erster Donnerstag im Monat, 19.30 Uhr*

# POLY-TEX

## The 17th Annual TEX Users Group Meeting

Dear TEX Friends!

So the time has arrived when TEX, the Polyglot, who has for many years happily "spoken" many different languages written in the Latin alphabet, extends its knowledge in the field of other alphabets. During the Summer of 1996, a visit is planned to Russia where TEX will have its first practical session in Cyrillic.

Russia is that large country, with its enormous spaces, inhabited by those enigmatic Russians, who started the century with a Revolution, and ended it with "Perestroyka". A country who founded a brilliant mathematical school, colonized the Cosmos, and conquered the world with its literature, music, and ballet.

So what awaits the TEX user who plans to attend TUG'96?

At Moscow's international Sheremetevo-2 Airport, conference participants will be met by a member of the organizing committee and escorted by bus to Dubna, where

from July 28 to August 2, the 17th TUG conference

will take place (see below for more about Dubna). If, for some reason or another, you envision cold, the taiga, and white bears, then we can reassure you, Dubna is in the European part of Russia, and if you want to see the ice on the nordic oceans, you will have to travel further than you would to reach the subtropics of the Black Sea shore. Indeed, you will soon find that the pine forest in which the comfortable town of Dubna is situated will remind you of a park. In July and August, the weather is mostly warm and sunny, with an average temperature of 28 degrees Centigrade. Guests will be housed in a comfortable hotel on the banks of the Volga river in single or double rooms (hot water, shower, telephone, and television). The Russian "cuisine" is characterized by its abundance, so one can forget about slimming.

The social program includes a picnic on the picturesque banks of the Volga, where we will be taken by boat, a bus excursion to Sergiev Posad (the center of the Russian Orthodox Church, where Andrey Slephkhin works—see his article on page 373 of TUGboat 16#4 or the EuroTEX'95 proceedings page 331), and, on the last day a visit to Moscow, following which the participants can be dropped of at the airport to fly home, or at one of the railway stations, if they want to prolong their visit.

The conference's preliminary program will be announced as soon as we receive from you, dear readers, proposals for presentations, courses that you would like to teach or attend, poster sessions, or any problem(s) or subject(s) that interest you. Please send your suggestions to the conference electronic address `TUG96@pds.jinr.ru`





The theme of the Conference is: PolyTEX, TEX or the art of multi-lingual, maths and technical typesetting.

### Program Committee

- Evgeniy Vasilievitch Pankratiev
  Moscow, Russia
  email: `pankrat@shade.msu.ru`
- Michel Goossens
  Geneva, Switzerland
  email: `goossens@cern.ch`
- Mimi Burbank
  Florida, USA
  email: `mimi@scri.fsu.edu`

### Deadlines

*Please find below the revised deadlines for submitting papers for TUG'96.*

As we want to have all articles in a final form ready to be printed by the time of the Conference, we need them by late June. If final revisions are not received by June 10, articles will not appear in the Proceedings Issue of TUGboat.

- Submission of abstracts — March 11 (new date)
- Acceptance signified to authors — March 20 (new date)
- Preliminary articles — April 12 (new date)
- Proposals for workshops, demos, and poster sessions — April 20
- Registration and transfer of a non-refundable sum of 100.–$ per person to a Moscow bank — May 31
- Visa supporting information — June 5
- Revised articles — June 10 (firm date)
- Start of Conference — July 28

### Organizing Committee

- Vladimir Vasilievitch Korenkov
  Dubna, Russia
  email: `korenkov@cv.jinr.ru`

- Irina Anatolievna Makhovaya
  Moscow, Russia
  email: irina@mir.msk.su
- Sebastian Rahtz
  Oxford, UK
  email: s.rahtz@elsevier.co.uk

## Practical Information

*Conference costs*

The Conference Committee foresees a cost in the range 550–600 $. This sum includes the complete cost of the conference, namely the registration fee, lodging (6 nights with six breakfasts, lunches, and dinners), coffee/tea breaks, social events, and transport from Sheremetevo Airport to Dubna. The payment should be made in the following way: a 'non-refundable' sum of 100 $ per person should be transferred to a Moscow bank account (see below) before June 1st. After receipt of that sum an official invitation, necessary for obtaining a visa (see below), will be faxed to the participant. The rest will be payable in cash upon arrival at the Conference (no credit cards or cheques can be used in Dubna). For participants of economically weaker countries or for students we hope to arrange partial support via sponsors. We hope to arrange bursary funds for support of students and those participants who demonstrate need.

*Bank Transfer Information*

Bank "Credit-Moscow"
Moscow, Russia 113054
6-TH Monetchikovsky Per., 8
Swift Code: CRMORUMM
A/C N 00107091799
(transfer rate: USD only)

*Visas*

Most of the visitors from outside Russia will need a visa to attend the Conference. Therefore, for arranging a visa into Russia, participants should inform

Mrs. N. Dokalenko, Conference Secretariat
email: nataly@ypr.jinr.dubna.su
Fax: 7/095/975 2381 or 7/09621/65 891

of their and (possibly) the accompanying person(s)'s full name, date of birth, citizenship, passport number, arrival and departure dates. The Secretariat will forward by fax the visa support message to the participants with which they should apply





for visas to the nearest Russian Embassy or Consulate. Please note that you should apply for a visa valid for Dubna, Moscow, and Sergiev Posad. (If you plan to visit other cities in Russia you should obtain the relevant documents and join them to the visa application, so that the names of all places to be visited can be entered on the visa form as required.)

*Transportation*

The Organizing Committee will arrange direct transportation by bus from the Sheremetevo-2 Airport to Dubna (130 km north of Moscow). The Secretariat should be informed of the flight number, precise date and time of arrival (Moscow time) 'no later than' four working days before a participant wishes to be met at the airport. It is our intention to have each participant met by a member of the Organizing Committee. Details will be available later.

## Welcome to Dubna

Dubna was founded in 1956 when the Convention establishing the Joint Institute for Nuclear Research (JINR) was signed. The town is situated on the picturesque banks of the Volga river and the Moscow sea 120 km to the north of Moscow. One can reach Dubna from Moscow within 2 hours going by car, by bus or by express train. It will take you 1.5 hours to go to Dubna from the Sheremetevo-2 International Airport. Waterways connect Dubna not only to the Russian Volga cities, but also to the waters of the Black, the Caspian, the Baltic and the White seas.

There is no harmful environmental impact from the industrial plants; this together with the large tracts of forest in the environs of Dubna, and the vast water area dotted with small islands, makes the area quite attractive for tourism and rest. The Volga embankment is one of the prettiest parts of the town. In springtime, the streets of Dubna are full of the odour of lilacs, the apple trees are pink-white; in summer, lime trees, maples, birch trees and poplars make the town seem totally green; in autumn the town is all golden excepting the evergreen of old pine trees. The town's modern look harmonizes with the quietness of the surrounding forest. The town was built in the midst of a forest. There are separate patches of trees in the town itself, and the town park is just a part of the forest. It takes just a few minutes to get to the forest from the shopping centre on foot. A few minutes' walk and you are outside the city limits!

Small as it is, Dubna is a real metropolis. It is a scientific metropolis. It is a "big little city", as a visiting American scientist called it many years ago. Since the foundation of the Joint Institute for Nuclear Research (JINR), the name of Dubna has constantly been in the pages of the world's newspapers and journals. Dubna is one of the world centres for fundamental research in nuclear physics. The Joint Institute plays an important role as a coordinator of investigations of the scientists

from 18 JINR member-state institutes. Wide international scientific and technical cooperation is one of the fundamental concepts of the JINR.

Dubna is indeed a town of international friendship. Foreign speech can be heard everywhere. But the words, no matter in which language they are pronounced, are clear to everybody: friendly cooperation and fraternity unite all the physicists and mathematicians living and working in Dubna into an international scientific community.

On October 3, 1994, Dubna opened its doors to its first university, "International University of Dubna: Nature, Society and Man". The university is composed of five "cathedra" or faculties, including socioeconomic sciences, ecology and earth science, computer education, linguistics, and health and physical education. Two more faculties—law and government and technology—are also being contemplated.

The town has great experience in holding international conferences, and exchanges of delegations between countries in the sphere of science, education and culture. Dubna and La Crosse, Wisconsin, USA, are sister cities. Dubna is famous for its hospitality. Famous scientists, public figures and statesmen from different countries visit Dubna. They are always impressed by the gracious welcome they receive in Dubna and warm generosity which Dubna residents demonstrate.

### A few Words on Moscow

Moscow, the capital of Russia, has a population of some nine million people. It is a city rich in cultural, architectural and historical monuments, and, at the same time, boasts a rapidly developing modern urban community with brand new blocks of flats, long, straight and broad avenues, parks, gardens, stadiums, schools, cinemas, department stores, recreation centres, bridges and highways. Though forward-looking, it cherishes the memory of its past, and its old sections lend it a special charm.

Michel Goossens
(TUG President)





# Adressen

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

Tel.: 0 62 21/2 97 66
Fax: 0 62 21/16 79 06
e-mail: `dante@dante.de`

Konten: Postgiroamt Karlsruhe
BLZ 660 100 75

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

### Präsidium

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

### Server

| | |
|---|---|
| ftp: | `ftp.dante.de` [129.206.100.192] |
| e-mail: | `ftpmail@dante.de` |
| gopher: | `gopher.dante.de` |
| WWW: | `http://www.dante.de/` |
| Mailbox: | 0 62 21/16 84 26 (nur für Mitglieder) |

# Autoren/Organisatoren

**Andreas Dafferner** [43]
Rohrbacherstr. 45
69115 Heidelberg

**Luzia Dietsche** [3, 43]
siehe Seite 60

**Rainer Hülse** [48]
Wilhelmstr. 67b
47807 Krefeld
Tel.: 02151/305884

**Joachim Lammarsch** [4]
siehe Seite 60

**Werner Lemberg** [30]
a7621gac@awiuni11.bitnet

**Irina Makhovaya** [55]
Executive Director CyrTUG
c/o MIR Publishers
2 Pervyi Rizhskii Pereulok
Moscow 129820, Rußland
Tel.: +7/095/286-0622
Fax: +7/095/288-9522
email: cyrtug@mir.msk.su

**Gerd Neugebauer** [6]
Mainzer Str. 8
56321 Rhens
gerd@informatik.uni-koblenz.de

**Espen S. Ore** [53]
Norwegian Computing Centre for the Humanities
Harald Haarfagresgt 31
5007 Bergen, Norwegen
Fax: +47/55/322656
allc-ach96-request@hd.uib.no

**Bernd Raichle** [15, 41, 43]
siehe Seite 62

**Volker RW Schaa** [43]
Riegerplatz 1
64289 Darmstadt

**Rainer Schöpf** [43]
siehe Seite 63

**Jolanta Szelatynska** [53]
GUST secretary
Ogolnouczelniany Osrodek Obliczeniowy UMK
ul. Chopina 12/18
87-100 Torun, Polen
mjsz@cc.uni.torun.pl

**Xerox Corporation** [53]
XSoft Division
3400 Hillview Avenue
Palo Alto, California 94 304, USA
ep96@xsoft.xerox.com





# Technischer Beirat

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

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

**Atari**
Stefan Lindner
Karolinenstr. 52b
90763 Fürth
atari@dante.de
*oder*
Lutz Birkhahn
Darfelder Str. 38
48727 Billerbeck
Tel.: 02543/4666
atari@dante.de

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

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

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

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

**Mailbox von DANTE e.V.**
Jürgen Unger
Ringstr. 24
64668 Rimbach
mailbox@dante.de

METAFONT
Jörg Knappen
Barbarossaring 43
55118 Mainz
metafont@dante.de

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

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

**PostScript**
Jürgen Glöckner
Ph.-Schmitt-Str. 8 b
69207 Sandhausen
Tel.: 06224/3750
postscript@dante.de

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

**Treiberentwicklung und SGML**
Joachim Schrod
Kranichweg 1
63322 Rödermark-Urberach
treiber@dante.de

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

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

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

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



## Inhalt Heft 4/1995