Die TEXnische Komödie

dante
Deutschsprachige
Anwendervereinigung TEX e.V.

21. Jahrgang Heft 1/2009 Februar 2009

1/2009

Impressum

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

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

Diese Ausgabe wurde mit `pdfTeX 3.1415926-1.40.9-2.2 (Web2C 7.5.7)` erstellt. Als Standard-Schriften kamen die Type-1-Fonts Latin-Modern und LuxiMono zum Einsatz.

Erscheinungsweise:	vierteljährlich		
Erscheinungsort:	Heidelberg		
Auflage:	2700		
Herausgeber:	DANTE, Deutschsprachige Anwendervereinigung TEX e.V. Postfach 10 18 40 69008 Heidelberg		
	E-Mail: `dante@dante.de` `dtkred@dante.de` (Redaktion)		
Druck:	Konrad Triltsch Print und digitale Medien GmbH Johannes-Gutenberg-Str. 1–3, 97199 Ochsenfurt-Hohe Stadt		
Redaktion:	Herbert Voß (verantwortlicher Redakteur)		
Mitarbeit :	Luzia Dietsche	Rudolf Herrmann	Gert Ingold
	Gerd Neugebauer	Rolf Niepraschk	Günter Partosch
	Bernd Raichle	Christine Römer	

Redaktionsschluss für Heft 2/2009: 15. April 2009 ISSN 1434-5897

Editorial

Liebe Leserinnen und Leser,

»Eins, Zwei, Drei« von Billy Wilder gilt als Klassiker der Filmbranche, wobei Sie sich hier sofort fragen werden, was dies mit unserer Zeitschrift »Die TEXnische Komödie« zu tun haben soll. Der Film kam mir auch nur in den Sinn, weil ich zufällig die Mitgliedsnummer 123 von Helmut Kopka sah, der Anfang dieses Jahres verstarb und im großen LATEX-Umfeld ebenfalls als »Klassiker« bezeichnet werden kann. »Der Lamport« war mein erstes LATEX-Buch, »der Kopka« mein zweites. Wirklich benutzt habe ich in meinen Anfangsjahren mit LATEX wohl nur ihn, wenn ich die äußeren Gebrauchsspuren betrachte. Eine Würdigung der Arbeit von Helmut Kopka finden Sie in diesem Heft.

Nachdem wir in der letzten Ausgabe unserer Zeitschrift »Die TEXnische Komödie« schwerpunktmäßig die Bibliografie zum Thema hatten, gibt es diesmal neben einem bunten Cocktail aus dem weitläufigen Anwendungsgebiet von TEX drei Beiträge zum Thema Fonts. Der erste ist die schriftliche Fassung eines Vortrages von der Tagung DANTE 2008 in Jena. Neben einer Beschreibung der Linux-Libertine finden Sie auch eine Anleitung zur Benutzung von Fonts, für die noch keine entsprechende LATEX-Anpassung vorliegt.

Klassische Komödien werden selten in der Originalfassung wiederaufgeführt; es gibt immer kleinere oder größere Änderungen. So auch bei unserer Vereinszeitschrift »Die TEXnische Komödie«, die nach langen Jahren ein wenig die äußere Form verändern möchte. Zwei Angebote für eine Überarbeitung des Layouts finden Sie auf Seite 77, wobei andere Vorschläge willkommen sind.

Ich wünsche Ihnen wie immer viel Spaß beim Lesen und verbleibe

mit TEXnischen Grüßen,

Ihr Herbert Voß

Hinter der Bühne

Vereinsinternes

Grußwort

Liebe Mitglieder,

wieder einmal ein Grußwort, so denken wir, wenn Herbert Voß zum Schreiben drängt, so denken aber vielleicht auch Sie, wenn Sie diesen Text an gewohnter Stelle in dieser Zeitschrift finden. Trotzdem ist dieses Grußwort etwas Besonderes, läutet es doch den 21. Jahrgang von »Die TEXnische Komödie« ein, ebenso wie das Jahr, in dem DANTE e.V. das 20-jährige Bestehen feiert. Wir werden sicher in der nächsten Ausgabe, die zeitlich näher an der Gründung im April 1989 liegt, auf dieses Thema zurückkommen.

Leider müssen wir mitteilen, dass unser Mitglied Helmut Kopka am 7. Januar verstorben ist. Er war als Buchautor zum Thema LATEX bekannt, insbesondere durch seine zum Schluss dreibändige Reihe, die als Standardwerk in deutscher Sprache gilt. Viele kannten ihn von den Tagungen – langjährige Mitglieder werden sich auch noch an »seine« Tagung in Katlenburg-Lindau erinnern. Sein Tod ist für die TEX-Gemeinde ein großer Verlust, aber er wird in unseren Gedanken noch lange weiterleben. Ihm zu Ehren finden Sie in dieser Ausgabe einen von Joachim Lammarsch verfassten Nachruf.

Während der Mitgliederversammlung in Tübingen wurde die Möglichkeit eingeführt, Beitragsrechung und Zuwendungsbescheinigung optional per E-Mail zu verschicken, da wir aus postalischen Gründen diese nicht mehr als kostengünstige Zeitschriftenbeilage verschicken dürfen. Wir werden in der nächsten Zeit die technische Grundlage hierfür schaffen. Meldungen für den elektronischen Versand nimmt das Büro entgegen (`office@dante.de`). Alle anderen Mitglieder werden Rechnung und Quittung am Jahresende per separater Post wie gewohnt auf Papier erhalten.

Mit freundlichem Gruß,

Klaus Höppner, Vorsitzender — Volker RW Schaa, Stellvertretender Vorsitzender

Helmut Kopka ist verstorben

Joachim Lammarsch

Helmut Kopka starb am 7. Januar 2009 im Alter von 76 Jahren. Den meisten ist er bekannt als Autor des dreibändigen Werks über LaTeX (*Einführung*, *Ergänzungen*, *Erweiterungen*) oder dem *Guide to LaTeX*, den er zusammen mit Patrick Daly verfasst hat. Er war auch ein langjähriges Mitglied von DANTE e.V., doch wie kann man ihn charakterisieren?

Als ehemaliger Vorsitzender hatte ich während meiner Amtszeit (1989–1998) Helmut ein wenig näher kennengelernt. Spontan fallen mir drei Gegebenheiten ein, die ihn gut beschreiben:

- 1994 hatten wir Probleme, einen Ort für unsere Mitgliederversammlung zu finden. Helmut bot sich uns sofort an, bei sich im Max-Planck-Institut für Aeronomie Domizil zu gewähren. Die, die dort waren, erinnern sich bestimmt noch an das von ihm am Abend organisierte Orgelkonzert von Prof. Vaselinuas.
- Das dreibändige Werk über LaTeX bestand anfangs nur aus zwei Bänden. Es waren auch die ersten Bücher, die in Deutsch über LaTeX verfasst wurden. Hier hat Helmut echte Pionierarbeit geleistet und wirklich sehr viel für die Verbreitung von LaTeX im deutschsprachigen Raum beigetragen. Allerdings waren die beiden Bände sehr umfangreich und viele Anfänger empfanden dies erst einmal als abschreckend. In Zusammenarbeit mit dem Lektor des Verlags Addison-Wesley entwickelte ich einen Vorschlag, die beiden Bände auf drei aufzuteilen, den Helmut auch begeistert umsetzte. Als das Ergebnis dann fertig war, waren es die drei Bände – wie man sie heute kennt – jeder einzelne aber genau so dick wie die früheren.
- Aber es war auch Helmut, der mich – ich glaube während der EuroTeX in Prag – beiseite nahm und mir sagte, er habe ein ganz schlechtes Gewissen, so viel Geld mit seinen LaTeX-Büchern zu verdienen. Auch mein Einwand, er habe die Arbeit gehabt, die Bücher zu schreiben, und er solle bedenken, wieviel er für die Verbreitung von LaTeX getan habe bzw. tun würde, konnte ihn nicht trösten. DANTE e.V. bekam eine sehr große Spende und ich eine Zusage, die ich bis heute nicht vergessen habe: *Joachim, wenn Du [der Verein] Hilfe oder Geld brauchst, dann lass es mich wissen.* Es war

ein generöses Angebot, auch wenn DANTE e.V. es niemals in Anspruch nehmen musste.

Helmut Kopka war bei den Ersten, die TEX aus den USA von der Texas A&M Universität mitbrachten und auf ihren Rechnern installierten. Mit dem Erscheinen von LATEX und danach LATEX 2_ε wechselte er zu diesem System, das er in seinen Büchern detailliert und präzise beschrieb. DANTE e.V. war für ihn immer eine wichtige Institution, die er unterstützte, wo er nur konnte. Ich bin mir sicher, wir werden noch lange an ihn denken. Um es mit Bertold Brecht zu sagen: Der Mensch ist erst wirklich tot, wenn niemand mehr an ihn denkt.

(Foto: Marion Lammarsch)

Bretter, die die Welt bedeuten

DEK-Verkehrsschrift mit METAFONT und LaTeX

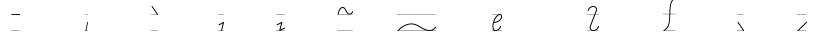
Stanislav Jan Šarman

Der Aufsatz präsentiert Text2DEK, eine METAFONT- und LaTeX-basierte Web-Anwendung[1], die deutsche Texte als Verkehrsschrift[2]-Stenogramme wiedergibt. Zuerst wird am Beispiel der DEK die Kurzschrift umrissen und danach geschildert, wie man in METAFONT[4] elementare Kurzschriftzeichen modelliert, deren Verbindungen untereinander zu Kurzschriftglyphen für Wörter (Steneme) in einer Metasprache beschreibt und sie als METAFONT-Zeichen implementiert. Eine Darstellung des Systemaufbaus und eine Zusammenfassung mit Seitenblick auf die Kurzschriftgeschichte und andere Kurzschriftsysteme schließen den Artikel ab.

DEK-Verkehrsschrift

Stenografie (Kurzschrift) ist im Vergleich zur »Langschrift« eine Raum und Zeit sparende sprachenspezifische Schreibschrift. Damit den Stenografen die Aufnahme der gesprochenen Rede in Echtzeit gelingt, werden in den von Buchstabenschriften ausgehenden Kurzschriftsystemen folgende Mittel der Verkürzung – hier am Beispiel der DEK illustriert – angewandt:

1. Rechtschreibregeln werden durch generelle Kleinschreibung, teilweise Ignorierung der Vokallänge, phonetisch-phonologische Schreibweise u. ä. gelockert: Beispielsweise werden *Ritt/ritt/riet* sämtlich als »rit« , *steht* als »stet«  und *Gesetz* als »gesez«  verschriftlicht.
2. Für die am häufigsten vorkommenden Wörter sind Kürzel vereinbart:



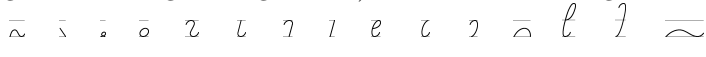
und die der ich in den ein das(s) mit von er es

[1] http://www3.rz.tu-clausthal.de/\char126rzsjs/steno/DEK.php

[2] Verkehrsschrift ist die Grundstufe der Deutschen Einheitskurzschrift (DEK).

Eine der ersten Untersuchungen zur Häufigkeit der Wortformen, Wortstämme, Präfixe, Suffixe, Silben und Buchstabenfolgen in der deutschen Sprache hatte mit einem Korpus von 11 Millionen Wörtern 1897–98 F. W. Kaeding vorgelegt. 1916 bemerkte J. B. Estoup, der wie Kaeding ein Stenograf war, dass das Vorkommen der häufigsten 20 bzw. 100 Wortformen etwa 1/3 bzw. 1/2 aller Wortformen im Sprachkorpus ausmacht. Diese Beobachtung, die uns verdeutlicht, wie die Kürzel die grafische Redundanz verringern können, führte später zum Zipfschen Gesetz, nach dem die häufigsten Lexeme, meistens Funktionswörter, auch die kürzesten sind. Da die Inhaltswörter länger sind, werden neben den Kürzeln noch andere Mittel der Verkürzung wie im nächsten Punkt dargestellt benutzt.

3. Die Zeichen für Konsonant(folgen), Präfixe und Suffixe (hier in der Reihenfolge ihrer Häufigkeit angeordnet) werden vereinfacht geschrieben:



n r l s m b g t d w h k f ch ein

Man beachte etwa, dass das Kurzschriftzeichen für »m« ◌ͮ als das letzte Drittel des Langschrift-»m« wiedergegeben wird. Generell werden in der DEK nur die Kurzschriftzeichen für Konsonant(folgen) ausgeschrieben. Vokale und Diphtonge dagegen werden durch einen die umgebenden Konsonanten verbindenden Aufstrich sinnbildlich dargestellt: Die relative Stellung und/oder Verstärkung[3] des nachfolgenden Konsonanten (»t« ◌ bzw. Punkt in Abbildung 1), bezogen auf die Basis des vorherigen Zeichens (hier »r« ◌), definiert den jeweiligen Vokal bzw. Diphthong, und zwar nach dem in Abbildung 1 gezeigten, *Auslautvokalisation* genannten Schema.

Elementare Kurzschriftzeichen

Zeichengewinnung

Aufgrund der Auslautvokalisation – die Zeichen für Vokale sind »nur« Verbindungsstriche – verstehen wir unter elementaren Kurzschriftzeichen die Zeichen für Konsonant(folgen), Präfixe und Suffixe. Ihre Form ist in handschriftlich geschriebenen, so genannten autografierten Vorlagen festgelegt [7]. Wir digitalisieren die normalerweise 15°–20° geneigten Zeichen nach einer vorherigen Rücktransformation in aufrecht stehende Zeichen. Im ersten Schritt werden, wie es bei der Generierung bzw. Offline-Erkennung von Handschriften üblich ist, die Koordinaten und Tangenten in Anfangs- und Endpunkten sowie in

[3] D. h. näher/weiter-, ganz hoch/hoch/neben/tiefgestellt und nicht/verstärkt (insgesamt zwölf Möglichkeiten für Vokale, zwei für »t«-Verbindungen).

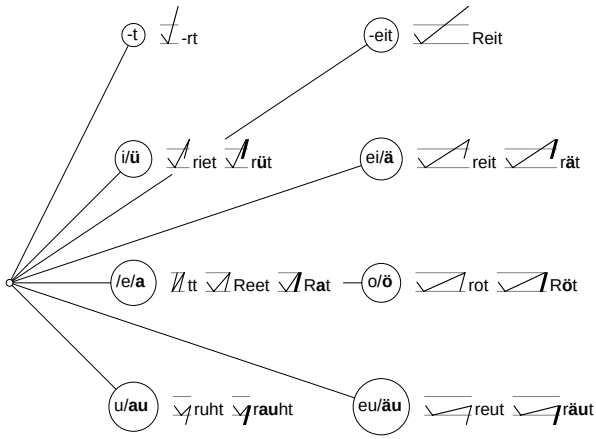


Abbildung 1: Schema der Auslautvokalisation

Punkten der lokalen Krümmungsextrema erfasst und in METAFONT-Pfaden (*paths*) verzeichnet: Kurven ohne Rückkehrpunkte sind dann in METAFONT automatisch tangentenstetig (*mock curvature continuous*). Der Vorgang sei am Beispiel des digitalisierten »d« ℓ veranschaulicht:

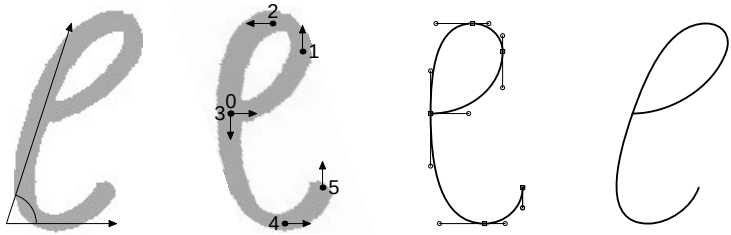


Abbildung 2: Digitalisierungsbeispiel

Tatsächlich ist das »d«, wie die meisten unserer Zeichen, mittels hermitescher Interpolation sogar als krümmungsstetige[4] Bézier-Spline-Kurve modelliert. Die

[4]Krümmungsstetige Kurven werden als G^2-stetige Kurven bezeichnet. Solche Kurven, nachdem sie in Bezug auf die Bogenlänge parametrisiert wurden, sind zweimal stetig differenzierbar, d. h. sie können mit stetiger Beschleunigung durchlaufen werden.

Aufgabenstellung der hermiteschen Interpolation für Bézier-Splines, nämlich aus gegebenen zwei Endpunkten $z_i (a = z_1 - z_0)$, Einheitstangenten d_i, ${d_i}{d_i}^t = 1$ und Krümmungen κ_i, $i = 0, 1$, die Kontrollpunkte $z_0^+ = z_0 + \delta_0 d_0$, $z_1^- = z_1 - \delta_1 d_1$ zu bestimmen, führt zum folgenden System von nichtlinearen Gleichungen [2]:

$$(d_0 \times d_1)\delta_0 = (a \times d_1) - \frac{3}{2}\kappa_1 \delta_1^2 \tag{1}$$

$$(d_0 \times d_1)\delta_1 = (d_0 \times a) - \frac{3}{2}\kappa_0 \delta_0^2 \tag{2}$$

Mit $a \times b := a_x b_y - a_y b_x$ bezeichnen wir das Kreuzprodukt zweier Vektoren in R^2. Die Gleichungen (1) und (2) sind nur in speziellen Fällen einfach lösbar, z. B. wenn eine der Krümmungen gleich 0 ist. Ein Beispiel hierfür ist das aus einer Geraden und einem Halbkreis bestehende Zeichen »b« ɓ. Es lässt sich G^2-stetig realisieren, indem das erste Achtelkreissegment durch ein Kreissegment ersetzt wird, in welchem die Krümmung im linken Endpunkt gleich 0 ist.

Ähnlich einfach ist der Fall, wenn die beiden Tangenten parallel sind, d. h. $(d_0 \times d_1) = 0$ und $\operatorname{sgn}(\kappa_i) = (-1)^i \operatorname{sgn}(d_i \times a)$: Bei der Modellierung von »m« ɯ kann das Anfangskreissegment mit dem Endkreissegment[5] durch hermitesche Interpolation krümmungsstetig verbunden werden, vgl. die beiden »m«-Varianten.

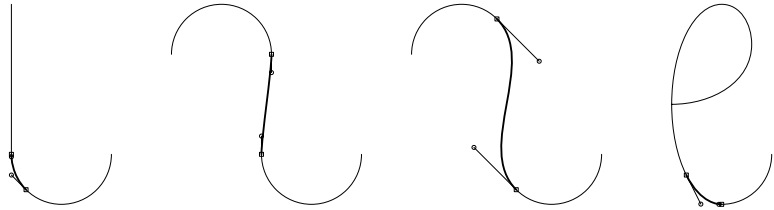


Abbildung 3: Modellierung von »b«, »m« und »d«

Die Formung von »d« ɗ ist komplizierter: Hier werden der Abwärtsteil des Zeichens analytisch als halbe Ellipse und der Rest als Viertelkreis modelliert. Beide Teile können durch Lösung der Gleichungen (1) und (2) G^2-stetig verbunden werden.

[5] G^2-stetige Kreissegmente sind in METAFONT als Makros vordefiniert, beispielsweise `halfcircle`.

Zeichenklassifizierung

Wir unterteilen die Zeichen nach der möglichen Strichrichtung der Tangente (abwärts/aufwärts) sowie positiver, negativer und verschwindender Krümmung an Anfangs- und Endpunkten in 5×5 rotationssymmetrische Typen, so z. B. das Zeichen *e*, welches zum Zeichentyp der 5. Zeile und 4. Spalte gehört: Es beginnt und endet in Aufstrichrichtung und die Krümmung ist durchweg nicht negativ (siehe Abbildung 4). Eine solche Klassifizierung ist eine Verfeinerung der in den kursiven Kurzschriften üblichen (hier spaltenweisen) Zeichenunterteilung in Geradeausläufe, Rechtsschräge, Linksausläufe, Rechtsrunde und Fußschleifen je nach der Endung.

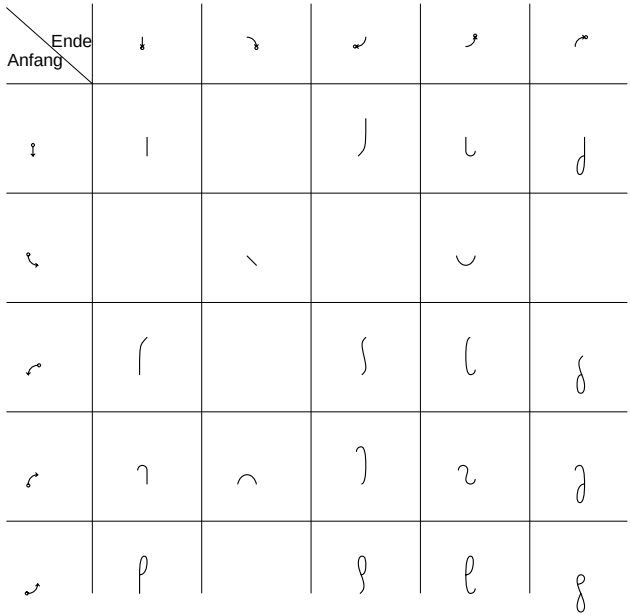


Abbildung 4: Zeichentypen

Die Auslautvokalisation eines vorhergehenden »a«, »ö«, … verlangt verstärkte Zeichenvarianten, die man beim Autografieren mit der Spitzfeder durch einen Schwellzug im Abstrichteil des Zeichens vollzieht. Die vom Zeichentyp abhängige Verstärkung erreicht ihr Maximum in Punkten mit lokal minimaler

Krümmung[6]. Anders als Züge nicht verstärkter Zeichen, die als `draw` mit dem Rundstift (*circular pen*) ausgeführt werden, modellieren wir die maximalen bzw. minimalen Zeichenbreiten mit `penpos` und zeichnen den verstärken Zeichenteil mit `fill`.

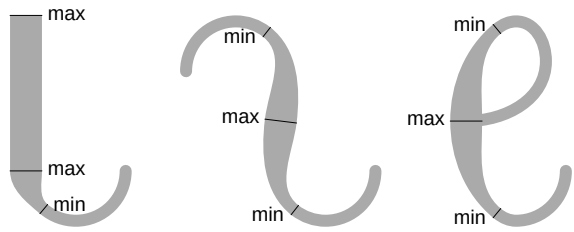


Abbildung 5: Verstärkte Zeichenvarianten

Die Zeichenverbindungen: Steneme

Metaform

Das Alternieren der Konsonanten und Vokale in einem Wort sowie das Auslautvokalisationsprinzip legen folgende allgemeine Metaform der DEK-Steneme, d. h. Kurzschriftglyphen für Wörter, nahe:

`(,`K`){(`V`,`K`)}`* Bsp. `(,t)(ü,r)` für das Wort »Tür«

K bezeichnet Konsonant- und V Vokalzeichen, d. h. ein Verkehrsschriftstenem besteht aus einem Konsonantzeichen, das mit beliebig vielen Vokalzeichen- und Konsonantzeichenpaaren verbunden wird.

Fehlende Konsonanten am Stenemanfang oder -ende werden durch einen Punkt, ein im Stenem fehlender Vokal wird durch ein Häkchen und ein fehlender Konsonant durch eine enge Verbindung realisiert.

[6]Das sind Punkte der lokal maximalen Geschwindigkeit.

Wort	*Metaform*	*Stenem*
Ritter	`(,r)(i,t)(e,r)`	
Ehre	`(e,r)(e,)`	
Tara	`(,t)(a,r)(a,)`	
Tiara	`(,t)(i,)(a,r)(a,)`	
Rottier	`(,r)(o,t)#(,t)(i,r)`	

Die von uns gewählte Metaform liefert unmittelbar das erzeugende Programm, z. B. wird die Metaform `(,r)(i,t)(e,r)` zu `I(r);J(i,t);J(e,r);` – einem zwischen `begin`/`endchar` eingebetteten METAFONT-Programmstück mit `I`(nit) und `J`(oin), also Initialisierungs- bzw. Verbindungsmakros. Die Kurzschriftglyphe für *Ritter* wird mit initialisiert, danach wird ein »t« und schließlich ein »r« angefügt. Der erste Parameter von `J` entscheidet über die relative Platzierung des im zweiten Parameter angegebenen Konsonantzeichens gemäß dem Auslautvokalisationsschema.

Die Zeichenverbindungen

Die Verbindungsroutine `J` unterscheidet die in Abbildung 6 dargestellten 5×5 Zeichenverbindungstypen, die sich aus der Klassifizierung nach dem Endtyp des linken bzw. nach dem Anfangstyp des rechten Zeichens ergeben:

Die Zeichen in dem linken oberen 2×2 Tabellenausschnitt werden Punkt zu Punkt verbunden. Singulär sind auch die Verbindungen in den Rückkehrpunkten des rechten Zeichens in der dritten Spalte. Hier wird die Verbindung iterativ so bestimmt, dass die Tangente im Rückkehrpunkt der Tangente der Verbindungsgeraden gleicht (vgl. »ow«). Für das linke bzw. rechte Zeichen in den beiden letzten Zeilen bzw. Spalten wird iterativ eine möglichst glatte Verbindung zwischen den nächst gelegenen Punkten mit waagerechter bzw. senkrechter Tangente und nicht verschwindener Krümmung gesucht.

Für die Verbindung des »b« mit »t« wie etwa in *Beet* wird auf $y = f(x)$ (dem letzten Viertelkreis von) ein Punkt $z = (x, y)$ mit der Eigenschaft $f'(x) = \frac{y_6 - f(x)}{x_6 - x}$ gesucht und zwar durch die Iterationsformel

$$f'(x^{(i+1)}) = \frac{y_6 - f(x^{(i)})}{x_6 - x^{(i)}}$$

oder äquivalent mit

$$x^{(i+1)} = f'^{-1}(\frac{y_6 - f(x^{(i)})}{x_6 - x^{(i)}}),$$

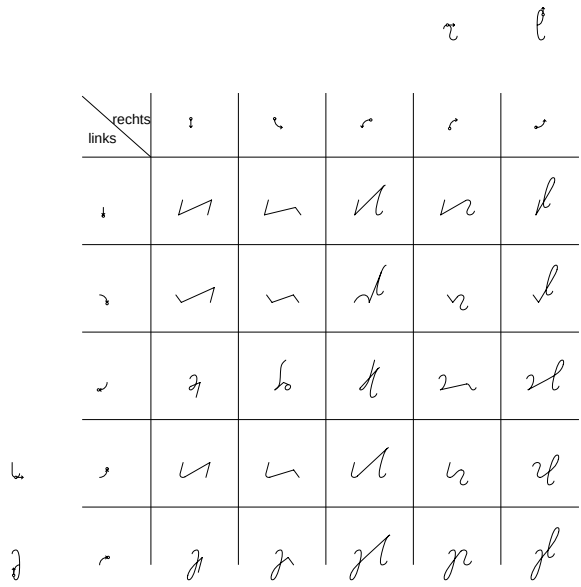


Abbildung 6: Zeichenverbindungstypen

da f'^{-1} der in METAFONT eingebauten Funktion `directiontime` gleicht und direkt auswertbar ist. Im Verbindungspunkt, hier $z^{(3)}$, lässt sich ähnlich wie bei der Konstruktion von ι im Punkte z_1 die Krümmung von der anderen Seite nachträglich so korrigieren, dass wir aus einer G^1- eine G^2-stetige Verbindung bekommen (vgl. Abbildung 7).[7].

Für die Verbindungstypen (4,4) und (5,5) liefert die hermitesche Interpolation wegen der Parallelität der Tangenten entsprechend den Gleichungen (1) und (2) unmittelbar krümmungsstetige Verbindungen, wie z. B. *nehm* ∿.

Text2DEK

Das entwickelte Onlinesystem wandelt Texte in DEK-Verkehrsschrift-Stenogramme um, z. B. den Satz von Franz Kafka:

`Vor dem Gesetz steht ein Türhüter.` →

[7] G^1 bzw. G^2-Stetigkeit ist affin, d. h. auch bei Scherung invariant.

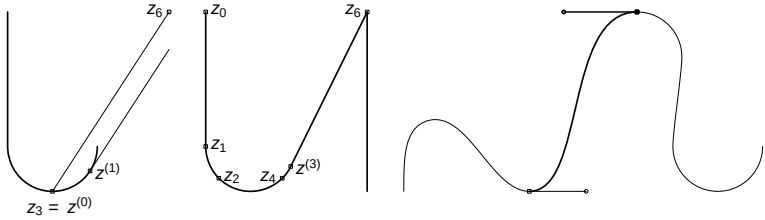


Abbildung 7: Das Korrigieren von Krümmungen

Das Stenogramm wird auf dem Webserver in folgenden Schritten erzeugt:

1. Der Eingabetext wird zuerst in einzelne, etwa eine Wortform umfassende, Tokens zerlegt.
2. Die Metaform eines Tokens wird entweder einem Wörterbuch entnommen (so bei Zahlen, Satzzeichen und Verkehrsschriftkürzeln) oder aus einer morphologischen Wortzerlegung regelbasiert ermittelt.
3. Anhand der Metaform wird für jedes Token *on the fly* in einem METAFONT-Lauf ein Stenem – ein METAFONT-Zeichen pro Wort – generiert. Die Glyphen werden in pk-Dateien abgelegt.
4. Der in Token zerlegte Text wird mit LaTeX und `dvips` übersetzt, bevor die Seiten des entstandenen Stenogramms als gif-Bilder zum Browser geschickt werden.

Übersichtstabelle zu unserem Beispiel:

Token	*Wortzerlegung*[8]	*Metaform*	*Stenem*
`.`		`(,_period_)`	
`Gesetz`	`ge^setz`	`(,ge)(,s)(e,z)`	
`Türhüter`	`tür#hüt~er`	`(,t)(ü,r)#(,h)(ü,t)(e,r)`	
`dem`		`(,dem)`	
`ein`	`ein`	`(,ein)`	
`steht`	`steh~t`	`(,st)(e,t)`	
`vor`		`(,vor)`	

Abbildung 8: Stufen der Stenogrammerzeugung

Tokenizer

Ein deterministischer Tokenizer versucht, nachdem er LaTeX-Befehle ausgesondert hat, zunächst Wort- und Satzenden richtig zu erkennen. Danach werden Zahlen, Satzzeichen, Abkürzungen und Verkehrsschriftkürzel von den anderen Wortformen getrennt. Abkürzungen (Akronyme und Alphabetismen) und Kürzel sind in lexc-Wörterbüchern [1] definiert. (Seltene) Homonyme mit verschiedenen Stenemen wie  vs.  werden durch ihre Nummer unterschieden: `meine` bzw. `meine#1` (Possessivpronomen) vs. `meine#2` (Verb in der 1. Person Singular).

Morphologische Analyse

Dieser Schritt der morphologischen Analyse ist notwendig, weil:

- gängige Präfixe wie `ge^`, `er^`, `zu^`, `ver^`, `vor^`, … und Suffixe wie `~ung`, `~heit`, `~keit`, … als elementare Kurzschriftzeichen realisiert sind:


`ge^näh~t`

`Er^trag`

`zu^ge^näh~t`

`Leit~ung`

`Gleich~heit`

`Über^heb~lich~keit`

- bei Komposita die Auslautvokalisation in der Regel unterbrochen wird und das Stenembild zur Grundlinie zurückkehrt,
  vgl. `tür#hüt~er`:  oder auch den Unterschied zwischen  und (`Fund#büro` und `Büro#fund`).

- Ligaturen an der Wortgrenze auszuschließen sind:
  Vgl. `Wach#stube`  vs. `Wachs#tube`.

Für die lokal in einem Cache gespeicherte morphologische Zerlegung wird der Lingsoftserver[9] konsultiert.

Als einer der ersten TeX-Anwender hat Yannis Haralambous [3] die morphologische Wortanalyse zur Unterscheidung des langen vom kurzen »s« beim automatischen Setzen deutscher Texte mit Fraktur- bzw. Sütterlinschrift[10] genutzt[11].

[8] Mit ^ und ~ bzw. # als Präfix-, Suffix- bzw. Kompositatrennzeichen; bei Kürzeln fehlt der Eintrag.

[9] `http://www2.lingsoft.fi/cgi-bin/gertwol`

[10] `http://www3.rz.tu-clausthal.de/~rzsjs/steno/Suetterlin.php`

[11] Auch Silbentrennungen mit TeX könnten von einer solchen Analyse profitieren [6]. Umgekehrt verwendet man den Befehl `\showhyphens` zur Silbenauszählung in stenografischen Diktaten.

Stenemizer

Die Aufgabe des Stenemizers ist es, die morphologische Wortzerlegung in die oben beschriebene Metaform zu überführen. Das Programm ist wie der Tokenizer in xfst – einem XEROX-*finite-state-morphology*-Tool [1] – geschrieben. Es besteht aus sukzessiv angewandten kontextabhängigen Regeln, angefangen mit `ie -> i, ph -> f, ä -> e, tz -> z`, d. h. einfachen Regeln, die die orthografische Schreibung durch eine phonologische Schreibweise ersetzen. Die nächste Regel

```
h -> 0 || VOC _ KON „ y -> ü || KON _ KON
```

eliminiert das »Dehnungs-h«, falls im linken Kontext ein Vokal und im rechten Kontext ein Konsonant steht. Gleichzeitig wird im Konsonantenkontext »y« durch »ü« ersetzt.

Zwei Regeln bilden den Stenemizer-Kern: Nach der ersten Regel werden zu Beginn Konsonanten und Vokale unterschiedlich geklammert, beispielsweise: `ver^gang~en~heit` → `[ver^][g](a)[ng]~(e)[n][~heit]`. Anschließend werden diese Klammern gemäß `"]["` → `")(","`, `"]("` → `")(","`, `")["` → `","(,"` `")("` → `",)("` zusammengefasst. So werden beispielsweise folgende Transformationen durchgeführt: `[ver^][g](a)[ng](e)[n][~heit]` → `[ver^)(,g)(a,ng)(e,n)(,~heit]` und `[t](i)(a)[r](a)` → `[t)(i,)(a,r)(a)`.

Diese an sich einfachen Regeln werden unter anderem dadurch kompliziert, dass es neben dem Abstrichzeichen für den Konsonanten »t« auch ein Aufstrichzeichen, das sogenannte End-»t« am Stammende nach Konsonanten gibt und das »t« auch in den Stammanlauten »st« und »str« vorkommt, die als einfache Zeichen geschrieben werden.

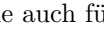
Text2DEK-Backend

Standardmäßig werden für die Webausgabe Bitmapfonts (PostScript Type 3) und experimentell auch Vektorfonts (PostScript Type 1) erzeugt. Ein `dvitype`-Clone kann außerdem eine nach Wortmaterial durchsuchbare djvu-Datei generieren, in der der unterliegende (Klar)text angezeigt wird, wenn die Maus über die Stenemglyphe fährt.

Caveat lector

Im Unterschied zu [5] geschieht die Kurzschriftgenerierung in unserem System automatisch. Wir stellen uns nicht die Frage, »ob Maschinen stenografieren können«. Text2DEK imitiert in einer Art von Turing-Test das Schreiben von Verkehrsschriftstenogrammen. Der kurzschriftkundige Leser wird wissen

wollen, wann die so generierten Steneme den Anforderungen von [7] nicht genügen. Dies tritt z. B. in den nachfolgenden Fällen auf:

- Wenn das System die richtige Aussprache oder morphologische Zerlegung eines Wortes nicht kennt: Man vergleiche die verschiedenen Ausprachen von »ie« in »riet«  bzw. »knien«  oder die Aussprache von »Ingenieur«  vs. »Ingeniör«  vs. »Inscheniör«  [12]. Wir konnten kein dem Unisys (siehe [8]) vergleichbares Aussprachewörterbuch für die deutsche Sprache in das System integrieren. Abgesehen davon kann `Text2DEK` Homographen wie »Móntage«  vs. »Montáge«  oder »Réntier«  vs. »Rentiér«  oder »LaTeX«  vs. »Latex«  nicht unterscheiden.
- Wenn [7] verlangt, dass, »um schwebende Zeichen zu vermeiden«, z. B. »dienen« als  oder »winken« als  und nicht als bzw. geschrieben wird. Unsere Verbindungsroutine kann nicht die kommenden Zeichen antizipieren.

Bei diesen Problemfällen sind Korrekturen von Menschenhand erforderlich und auch möglich.

Cave canem

Das abgebildete, spätestens 79 n. Chr. entstandene Pompeji-Mosaik führt uns eindrucksvoll den Schritt von einem Ideogramm zur römischen Antiqua vor.

Zur gleichen Zeit gab es als Schreibschrift die ältere römische Kursive (genauso eine Buchstabenschrift), aber auch die Tironischen Noten, eines der ersten stenografischen Systeme überhaupt, allerdings mit Symbolen sowohl für Wörter wie auch für Silben und Buchstaben.[13]

In Abbildung 9 ist dieser Schriftzug auch in den aktuellen Stenografien geschrieben. Alle Systeme fußen auf Buchstabenschriften, benutzen allerdings phonologische Schreibweisen. Vokale werden in diesen Systemen unterschiedlich dargestellt: Die deutsche DEK benutzt durchweg Auslautvokalisation; die tschechische/slowakische Herout-Mikulík-Kurzschrift symbolisiert das »a« aber durch Anlautvokalisation. Bei dem amerikanischen Gregg-System [8] werden Vokale als Ellipsen ausgeschrieben, beim englischen Pitman-System

[12] DEK hält hier »In« für ein Präfix.
[13] Die Tironischen Noten verdanke ich M. Hellmann.

Abbildung 9: Weg vom Ideogramm über römische Antiqua zur Kurzschrift.

werden sie durch diakritische Zeichen und Stenemstellung in Bezug auf die Grundlinie dargestellt.

Um Steneme etwa in METAFONT beschreiben zu können, braucht man unterschiedliche Metaformen und auch unterschiedliche Routinen zur Konsonantenverbindung. Auffallend auch im Vergleich zur Langschrift ist die flüssige Schreibweise der Stenogramme, die die Kodierung und Glyphengenerierung für einzelne Wörter impliziert, nicht nur für einzelne Buchstaben, wie es sonst in Satzsystemen der Fall ist.

Literatur

[1] Kenneth R. Beesley und Lauri Karttunen: *Finite State Morphology*; CSLI Publications; Stanford; 2003.

[2] Carl de Boor, Klaus Höllig und Malcom Sabin: *High accuracy geometric Hermite interpolation*; *Computer Aided Geometric Design*; 4, S. 269–278; 1987.

[3] Yannis Haralambous: *Drucksatz in gebrochenen Schriften (»Ten years after«)*; 2000; `http://omega.enstb.org/yannis/pdf/dante2000.pdf`.

[4] Donald E. Knuth: *The METAFONT book*; Bd. C von *Computers and Typesetting*; Addison-Wesley Publishing Company; Reading, Mass.; 5. Aufl.; 1990.

[5] Matthias Kuhn: *Rechenerbasierte Generierung von Stenografik*; Universität Ulm; 2003; Diplomarbeit.

[6] Christine Römer und Herbert Voß: *Deutsche Silbentrennmuster*; 2008; Vortrag Dante 2008 in Jena, `http://www.personal.uni-jena.de/~xcr/v2/?Seite=78`.

[7] Hans Treschwig (Hg.): *Systemurkunde der Deutschen Einheitskurzschrift, Wiener Urkunde*; Heckners Verlag; Wolfenbüttel; 1968.

[8] Stanislav Jan Šarman: *Writing Gregg Shorthand with METAFONT and LaTeX*; *TUGboat, TUG 2008 Conference Procceedings*; 29(3), S. 458–461; 2008.

TrueType-Fonts für pdfTEX

Hàn Thế Thành

Das mit TEX meistverwendete Vektorformat für Fonts ist Type 1. TrueType-Schriften (TTF) unterscheiden sich leicht von Type 1-Schriften. Ihre Verwendung erfordert daher etwas mehr Aufwand. Dabei ist es hilfreich zu verstehen, wie bei Type 1 und TTF die Benennung und Kodierung von Glyphen erfolgt – oder genauer: die »Identifikation« von Glyphen.

Wir betrachten zuerst Type 1, das TEX-Benutzern vertraut ist: Type 1 identifiziert Glyphen durch Bezeichner wie `/A`, `/comma`, usw. Anhand eines Glyphenbezeichners kann leicht festgestellt werden, ob eine Type 1-Schrift diese Glyphe enthält. Die Zeichensatzkodierung ist bei Type 1 daher sehr einfach: Zahlen zwischen 0 und 255 werden durch die Zeichensatzkodierung auf den Bezeichner der entsprechenden Glyphe abgebildet.

Bei TTF ist die Sache nicht so einfach, da an Stelle von Bezeichnern »Indizes« zum Verweis auf Glyphen verwendet werden. Jede Glyphe wird durch ihren Index statt durch einen Bezeichner identifiziert. Indizes sind Nummern, die sich von Schrift zu Schrift unterscheiden. Die Zeichenkodierung erfolgt bei TTF durch die so genannte »cmap«. Das ist vereinfacht gesagt eine Tabelle, die Zeichennummern auf Glyphenindizes abbildet. Eine TTF-Schrift kann eine oder mehrere solcher Tabellen enthalten (jede Tabelle entspricht dabei einer Kodierung).

Glyphenbezeichner sind in TTF-Schriften nicht immer vorhanden, da sie nicht zwingend erforderlich sind. Folgende Fälle sind für eine TTF-Schrift möglich:

- Die Schrift enthält korrekte Bezeichner für alle Glyphen. Dies ist der Idealfall und für viele hochwertige lateinische Schriften erfüllt.
- Die Schrift enthält falsche Bezeichner für alle oder viele Glyphen. Dies ist der ungünstigste Fall, der oft bei minderwertigen oder transformierten Schriften auftritt.
- Die Schrift enthält keine Glyphenbezeichner. Ein Beispiel hierfür sind neuere Versionen der Linotype-Palatino-Schriftschnitte, beispielsweise die bei Windows XP mitgelieferte Version 1.40.

[1] Übersetzt von Jan Steffan.

- Die Schrift enthält korrekte Bezeichner für die meisten Glyphen und sonst keine oder nur wenige falsche Bezeichner. Dies kommt gelegentlich vor.

Der Ursache für die komplexe Situation in Bezug auf Glyphenbezeichner bei TTF ist, dass TTF-Schriften im Gegensatz zu Type 1-Schriften auch ohne korrekte Glyphenbezeichner einwandfrei funktionieren. Hat dagegen eine Glyphe in einer Type 1-Schrift einen falschen Bezeichner, so fällt dies sofort auf. TTF-Schriften verwenden, wie erwähnt, keine Glyphenbezeichner zur Zeichenkodierung, so dass falsche oder fehlende Bezeichner in der Regel keine Probleme verursachen und nicht bemerkt werden.

Die Schwierigkeit bei der Verwendung von TTF-Schriften mit pdfTEX liegt darin, dass wir so stark mit der Type 1-Konvention der Glyphenbezeichner verhaftet sind. Die meisten Werkzeuge zur Handhabung von Schriftdateien erwarten diese Konvention; alle Encoding-Dateien (Endung `.enc`) verwenden ebenfalls Glyphenbezeichner. Wie oben diskutiert, können wir uns bei TTF-Schriften nicht auf die Glyphenbezeichner verlassen. Schriften, die nicht für alle Glyphen korrekte Bezeichner enthalten, erfordern eine Sonderbehandlung.

Wenn die Glyphenbezeichner falsch sind, benötigen wir eine bessere Alternative zur Identifikation von Glyphen in TTF-Schriften. Der verlässlichste Weg scheint über Unicode zu führen: Die meisten TTF-Schriften enthalten eine korrekte Abbildung von Unicode-Zeichen auf Glyphenindizes. Hierauf können wir uns verlassen, da TTF-Schriften ohne diese Abbildung nicht benutzbar sind.

Ab Version 1.21a unterstützt pdfTEX in Encoding-Dateien Bezeichner der Form `uni`*`XXX`*. Dies ist natürlich nur in Verbindung mit TTF-Schriften sinnvoll. Wenn pdfTEX beispielsweise den Bezeichner `uni12AB` einliest, führt es folgende Schritte durch:

- Einlesen der Tabelle ⟨unicode⟩ → ⟨glyph-index⟩ aus der Schriftdatei;
- Aufsuchen des Eintrags 12AB in der Tabelle; falls vorhanden, Verwendung des entsprechenden Glyphenindex.

`ttf2afm` wendet für Bezeichner der Form `uni12AB` das selbe Verfahren an.

Um eine TTF-Schrift mit pdfTEX verwenden zu können, sind mindestens die folgenden Schritte erforderlich:

- Erzeugen einer `.afm`-Datei aus der `.ttf`-Datei mithilfe von `ttf2afm`. Beispiel: `ttf2afm -e 8r.enc -o times.afm times.ttf`

- Umwandeln der `.afm`- in eine `.tfm`-Datei durch ein geeignetes Werkzeug (`afm2tfm`, `fontinst`, `afm2pl`, usw.)
  Beispiel: `afm2tfm times.afm -T 8r.enc`
- Anlegen des map-Eintrags für die Schrift.
  Beispiel: `\pdfmapfile{+times TimesNewRomanPSMT <8r.enc <times.ttf}`

Dies genügt für den einfachen Fall, in dem korrekte Glyphenbezeichner vorhanden sind. Nun betrachten wir eine Schrift, die keine brauchbaren Glyphenbezeichner enthält, beispielsweise Linotype Palatino in Version 1.40. Wir gehen davon aus, dass die T1-Kodierung verwendet werden soll. Zuerst kopieren wir die Dateien `pala.ttf` und `ec.enc` in das Arbeitsverzeichnis.

Ein erster Ansatz zur Konvertierung wäre dann
`ttf2afm -e ec.enc -o pala.afm pala.ttf`.

Da jedoch die Bezeichner aus `ec.enc` in `pala.ttf` nicht vorhanden sind (diese Schrift enthält gar keine Bezeichner), würden wir eine Reihe von Warnmeldungen erhalten:

```
Warning: ttf2afm (file pala.ttf): no names available in 'post' table, print
glyph names as indices

Warning: ttf2afm (file pala.ttf): glyph 'grave' not found
...
```

Die erzeugte Datei `pala.afm` würde gar keine Bezeichner enthalten. An Stelle der Glyphenbezeichner aus `ec.enc` erhalten wir seltsame Bezeichner wie `index123`. Die Kodierung enthält keine einzige Glyphe:

```
C -1 ; WX 832 ; N index10 ; B 24 -3 807 689 ;
...
```

Beim zweiten Versuch wird dann keine Kodierung angegeben:
`ttf2afm -o pala.afm pala.ttf`

Diesmal erhalten wir weniger Warnmeldungen, da wir `ttf2afm` nicht aufgefordert haben, die erzeugte `.afm`-Datei umzukodieren:

```
Warning: ttf2afm (file pala.ttf): no names available in 'post' table, print
glyph names as indices
```

Die `.afm`-Datei ist jedoch mit dem Ergebnis des ersten Versuchs identisch. Das hilft uns nicht weiter, da wir mit Bezeichnern der Form `index123` wenig anfangen können.

Daher versuchen wir es mit Unicode:

```
ttf2afm -u -o pala.afm pala.ttf
```

Diesmal erhalten wir andere Warnmeldungen der folgenden Art:

```
Warning: ttf2afm (file pala.ttf): glyph 108 have multiple encodings (the
first one being used): uni0162 uni021A
...
```

Es ist schwer, auf Anhieb zu verstehen, was uns `tfm2afm` mit der Warnmeldung mitteilen will. Daher rufen wir uns den Zusammenhang zwischen Glyphenbezeichner, Glyphenindex und Unicode in Erinnerung:

- Intern werden Glyphen durch einen Index identifiziert.
- Die Abbildung ⟨Glyphenbezeichner⟩ → ⟨Glyphenindex⟩ ist optional und nicht immer verlässlich. Die gleiche Aussage gilt für ⟨Glyphenindex⟩ → ⟨Glyphenbezeichner⟩.
- ⟨Unicode⟩ → ⟨Glyphenindex⟩ ist fast immer vorhanden und verlässlich.
- ⟨Glyphenindex⟩ → ⟨Unicode⟩ ist nicht immer verlässlich und keine Abbildung, da mehrere Unicode-Zeichen auf den selben Glyphenindex abgebildet werden können.

  Daher ist es ausgehend von einem Glyphenindex nicht immer möglich, ein korrespondierendes Unicode-Zeichen zu finden: Es kann sowohl keines als auch mehr als eines geben. Gibt es keines, so wird der Glyphenindex verwendet (z. B. `index123`). Im obigen Fall werden die Unicode-Zeichen 0162 und 021A auf den Glyphenindex 108 abgebildet. Da `tfm2afm` Glyphen auf Unicode-Zeichen abbilden soll, ist keine eindeutige Entscheidung möglich. `tfm2afm` wählt schlicht das erste Unicode-Zeichen und gibt eine entsprechende Warnmeldung aus.

Wenn wir nichts weiter wollen als `pala.ttf` mit der T1-Kodierung zu verwenden (und auf Ligaturen verzichten), ist der vermutlich einfachste Weg, eine neue `.enc`-Datei `ec-uni.enc` von `ec.enc` abzuleiten, in der alle Glyphenbezeichner durch Unicode-Bezeichner ersetzt sind. Das kann z. B. leicht durch ein Perl-Script erledigt werden, das dazu die AGL (Adobe Glyph List)[2] einliest. Ausgehend von `ec-uni.enc` können die nächsten Schritte zur Erzeugung der `.tfm`-Datei so aussehen:

```
ttf2afm -u -e ec-uni.enc -o pala-t1.afm pala.ttf
afm2pl pala-t1.afm
pltotf pala-t1.pl
```

[2] Verfügbar unter `http://www.adobe.com/devnet/opentype/archives/glyphlist.txt`.

Die Schrift kann dann folgendermaßen verwendet werden:

```
\pdfmapline{+pala-t1 <ec-uni.enc <pala.ttf}
\font\f=pala-t1\f
Dies ist ein Test der Schrift Palatino Regular in T1-Kodierung.
```

Wenn wir mehr wollen als die Benutzung von `pala.ttf` mit der T1-Kodierung, beispielsweise die Weiterverarbeitung der `.afm`-Datei mit `fontinst`, um kompliziertere Schrifteigenschaften nutzen zu können, dann müssen wir etwas anders vorgehen. Eine `.afm`-Datei, bei der alle Glyphenbezeichner in das `uni`*XXX*-Format überführt wurden, ist mit `fontinst` kaum verwendbar. Stattdessen benötigen wir eine `.afm`-Datei mit AGL-Bezeichnern. Diese erzeugen wir folgendermaßen:

- Erzeugen einer `.afm`-Datei mit Glyphenbezeichnern der Form `uni`*XXX*: `ttf2afm -u -o pala.afm pala.ttf`
- Umwandeln von `pala.afm` in `pala-agl.afm`, so dass `pala-agl.afm` nur AGL-Bezeichner enthält. Auch dies kann ein einfaches Perl-Script erledigen.
- Verarbeitung von `pala-agl.afm` mit `fontinst`.
- Wenn die `.tfm`-Dateien aus `fontinst` und Co und die map-Einträge (mit `fontinst` oder manuell erstellt) vorliegen, müssen in einem letzten Schritt die Kodierungen durch ihr `uni`*XXX*-Gegenstück ersetzt werden. Verlangt `fontinst` etwa das Hinzufügen der Zeile `pala-agl-8r <8r.enc <pala.ttf` zur map-Datei, so muss diese durch `pala-agl-8r <8r-uni.enc <pala.ttf` ersetzt werden, wobei `8r-uni.enc` durch Umsetzen der Glyphenbezeichner in die `uni`*XXX*-Form von `8r.enc` abgeleitet wird.

Der Font »Linux Libertine« und XeTeX

Philipp H. Poll, Michael Niedermair

Dieser Artikel zeigt in einem geschichtlichen Abriss, wie der Font »Linux Libertine« entstanden ist, welche Gedanken, Ideen usw. eingeflossen sind und welche Möglichkeiten dieser unter XeTeX entfaltet.

Die geschichtliche Entwicklung

von *Philipp H. Poll*

Die Geschichte der »Linux Libertine« beginnt im Sommer 2003 in einem etwas ranzigen Berlin-Kreuzberger Café. Wegen des akuten Mangels an freien Schriften, besonders solchen in progressivem Format, hatte ich mir überlegt, eine moderne, vielseitig verwendbare Schrift für die OpenSource-Gemeinschaft zu entwickeln. Nun saß ich mit meinem karierten Umweltpapier im besagten Café und versuchte mich an den Proportionen. Von der kosmopolitischen Times wollte ich die neue Schrift absetzen. Sie sollte keinen derart starken Fett-Fein-Kontrast besitzen, etwas eleganter sein und im Schriftbild nicht so gedrungen, denn dies sind meiner Überzeugung nach die Nachteile der Times, die einst für den Zeitungsdruck auf fleckigem, einfachen Papier entworfen wurde. Die »Linux Libertine« sollte hingegen freundlicher, leichter und auch für längere Zeilen tauglich sein. Daher wollte ich eine praktikable Mischung aus den edlen Buchschriften mit ihren großen Ober- und Unterlängen auf der einen Seite und den aktuellen Computer-Fonts auf der anderen Seite. Bedingt durch die geringe Auflösung der Computerbildschirme und den Wunsch der Entwickler nach perfekter Darstellung am PC, sind die Betriebssystemschriften heute mehr oder minder an ihrer organischen Form amputiert oder anders ausgedrückt, über das Maß vereinfacht. In Abbildung 1 sind zum Vergleich sechs Schriftproben dargestellt. Zuoberst [1] Times New Roman. Es folgen weitere ausgeprägte Bildschirmschriften mit kantigen Serifen und vereinfachtem unorganischem Bild: [2] DejaVu Serif, [3] Georgia, [4] Liberation Serif. Dagegen die klassische, organisch geformte Janson [5] mit abgerundeten, gekurvten Serifen und ausgeprägter Ober- und Unterlänge. [6] »Linux Libertine«.

Während ich also im Tee rührte, zeichnete ich die ersten Entwürfe für die Kleinbuchstaben – zunächst noch mit der Idee einer stärkeren Eigengestalt.

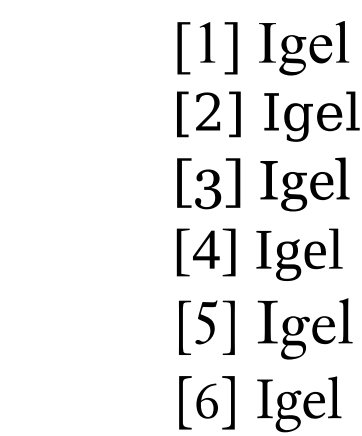


Abbildung 1: Schriftproben

Jedoch wurde mir mit der Zeit bewusst, dass die besseren Brotschriften eben jene unauffälligen sind, also solche, über die der Leser hinwegliest, ohne sich an einer ungewohnten Form zu stoßen. Auch auf die Gefahr hin, der Anwender könne sie mit anderen Schriften zu leicht verwechseln, wollte ich die »Linux Libertine« nun mit Konzentration auf die Lesbarkeit entwickeln. Zu dem Zweck studierte ich die Janson, Garamond, Minion, ComputerModern und viele weitere Schriften und fertigte eine Tabelle zu ihren Proportionen an. Für die technische Umsetzung war ein überraschendes Maß an Kenntnis über OpenType- und TrueType-Interna vonnöten, sodass ich Handbücher und Standardisierungsbeschreibungen lesen musste. Die zeichnerische Gestaltung hingegen war weit aufwändiger. Mehrfach musste ich nach Verbesserung der Anstriche alle Serifen der bisher gezeichneten Buchstaben austauschen. Und der Zeichensatz war bereits groß, trotzdem wollte ich die neue Schrift von Anfang an mit allen europäischen Sonderzeichen ausstatten. Unicode-Standards waren zu studieren und stundenlang Lettern zu zeichnen … Wie sieht ein perfektes „g“ oder „Ж“ aus? Manche Formen waren zickig und wollten sich nur über monatelanges Korrigieren langsam entwickeln. Oft zeichnete ich bis in die Nacht. Nach getaner Arbeit druckte ich die jeweils letzte Version in Form einer einseitigen Kurzgeschichte aus, um sie anderntags mit auf den Weg zur Uni zu nehmen. Während ich also morgens in der Berliner U-Bahn-Linie 1 von Kreuzberg nach Dahlem fuhr, saß ich mit Lupe und Bleistift

gewappnet über dem Korrekturblatt. Im Vergleich mit älteren und neueren Drucksachen studierte ich die nötige Serifenstärke, Zeilenabstand, x-Höhe und Gesamterscheinung. Die schwierigste Arbeit war es dabei, den richtigen Abstand zwischen Buchstabe und Folgebuchstabe zu ermitteln. Das sogenannte Ausgleichen und dabei die passende Laufweite zu finden, kostete die meiste Zeit. In meinem Beispieltext suchte ich das Vorkommen des jeweiligen Buchstabens und kontrollierte den Abstand zwischen allen möglichen Kombinationen. Mit einfachen oder doppelten Plus- und Minuszeichen kennzeichnete ich Verbesserungsbedarf.

Die U-Bahn brauchte für die über 100 Jahre alte Strecke fast genau eine halbe Stunde und schaukelte dabei nicht selten so sehr, dass ich die Korrekturen im Zeitraum des Halts nachtrug. Trotz dieser Beeinträchtigung konnte ich auf diese Weise eine Stunde pro Tag in die Revision des Schriftbildes stecken. Um zu testen, wie die »Linux Libertine« auf professionellen Druckern aussah, verwendete ich bei Gelegenheit Institutsgeräte. Einmal machte ich auch heimlich einen Ausdruck auf einem PostScript-Farblaser, dessen Nutzung uns Studenten aufgrund der vergleichsweise hohen Kosten eigentlich nicht gestattet war. Gerade dieser Druck brachte jedoch eine neue Erkenntnis für die korrekte Fonterstellung aus FontForge heraus. Alle zuvor verwendeten Drucker kamen gut mit der Verwendung von Verknüpfungen innerhalb des Fonts klar. Ein »ö« bestand beispielsweise aus der o-Verknüpfung und einer solchen zu dem Doppelpunktakzent. Auf dem besagten Drucker verschoben sich solche kombinierten Zeichenteile und erzeugten hässlich verunstaltete Ausreißer. Seither bestehen alle akzentuierten Zeichen aus eigenen Objekten. Dadurch verdoppelt sich praktisch die Größe einer Fontdatei, aber was will man machen, wenn die Druckerhersteller ihre Treiber nicht sauber implementieren?

Von Anfang des Projekts an war es immer auch mein Wunsch gewesen, kollaborativ zu arbeiten. Da eine Schrifttype konsequent gleichmäßig durchgestaltet sein muss und nur wenige Entwickler bemüht sind, sich in den Themenkomplex der digitalen Typografie hineinzuarbeiten, habe ich stets die grafische Entwicklung und Einarbeitung übernommen. Michael Niedermair hat sich seinerseits auf die Einbindung der »Linux Libertine« in LaTeX konzentriert. Von vielen weiteren Seiten kamen des Weiteren – teils direkt per Mail, teils über den Bug-Tracker – wichtige und gute Hinweise zur Verbesserung einzelner oder sogar ganzer Zeichengruppen. So verfügt die »Linux Libertine« heute über einen besonders guten kyrillischen Zeichensatz. Mir sei diese kurze Ausschweifung in die Welt der kyrillischen Lettern verziehen, denn sie zeigt beispielhaft, in welchen Bereichen kollaborative OpenSource-Arbeit besonders

erfolgreich sein kann. Beim Gestalten dieser Zeichengruppe musste ich nämlich lernen, dass die kyrillischen Buchstaben von den meisten Schriftentwicklern nicht nur oft vergessen werden (obwohl es mehr als 200 Millionen Nutzer gibt). So sorgt kurioserweise die Dominanz amerikanischer Software-Unternehmen dafür, dass gerade die Entwickler für breit angelegte Unicode-Schriften kaum Erfahrungen mit Sonderzeichen und fremdländischen Zeichensätzen haben. Ein russischer Typograf lehrte mich, dass die Erwartung kyrillischer Leser an Form und Proportion der Buchstaben völlig verschieden von der westeuropäischen ist und beispielsweise die Unicode-Tabellen oder Fonts wie Times New Roman kein Beispiel für gute kyrillische Typografie sind. Lange studierte ich alte russische Schriften und korrespondierte mit Muttersprachlern, bis die heutige Form gefunden war.

Viel ließe sich noch sagen über die schrittweisen Verbesserungen und Entwicklungen einzelner Buchstaben oder des gesamten Erscheinungsbildes der »Linux Libertine«. Zu viel für eine kleine Abhandlung wie diese, weshalb ich Interessierte auf unsere Internetseite verweisen möchte. Nur so viel noch: Inzwischen weist die »Linux Libertine« weit mehr als 2000 Zeichen auf. Ein besonderes Augenmerk galt der Unicode-Vollabdeckung westlicher Zeichensätze und aktueller computertypografischer Technik wie OpenType, welche es auch Laien ermöglicht, mit der entsprechenden Software wie XeTeX professionell zu setzen.

Die Zahlen

Aus typografischer Sicht sind Zahlen im Fließtext problematisch, denn die Form der arabischen Ziffern weicht stark vom Erscheinungsbild der lateinischen Buchstaben ab, sodass die Zahlen schließlich optisch aus dem Text hervorstechen. Da besonders am Rechner verfasste Texte gewöhnlich sehr zahlenlastig sind, verwendet die »Linux Libertine« mehrere Kniffe für einen optischen Ausgleich. Beispielsweise sind die Standardziffern der »Linux Libertine« niedriger als die Klein- und die Großbuchstaben, wodurch sie sich besser in das Buchstabenschriftbild einfügen. Währungssymbole wie das Eurozeichen oder Grad-Celsius sind ebenfalls auf die »Zahlen« verkleinert, sodass typische Preisangaben, wie „1.800€“, im Fließtext weniger dominant sind. Für besonders elegante Texte hält die »Linux Libertine« des Weiteren einen Satz an so genannten Mediävalziffern bereit. Diese sind untereinander vertikal so verschoben, dass sie teilweise auch Unterlängen aufweisen und dadurch den lateinischen Buchstaben ähnlicher werden. Im Buchdruck werden Mediävalziffern daher gerne benutzt. Beispielsweise in Fällen wie:

„Am 15. Juli 1606 wurde Rembrandt im niederländischen Leiden geboren.“

Beide Ziffernsätze, den Normal- und den Mediävalsatz, gibt es jeweils in doppelter Ausführung. Den Unterschied zwischen beiden Formen kann man in der Zeichentabelle nicht erkennen. Es geht hierbei nicht um die Form der Ziffer an sich, sondern um ihr »Fleisch«, d. h. den Abstand links und rechts von ihr, welcher sie von der vorigen bzw. nächsten Ziffer trennt. Der Standardziffernsatz ist für gewöhnlich – und so auch in der »Linux Libertine« – der Tabellenziffernsatz. Jede Ziffer, egal ob breit (wie die 8) oder schlank (wie die 1), hat dabei eine Einheitsbreite. Sinn dieser Einstellung ist es, dass mehrziffrige Zahlen in Tabellen sauber übereinander stehen. Sie sind also *dicktengleich* oder, wie durch die verbreitete Anglizismisierung des Deutschen heute gesagt würde, *monospaced*. In allen anderen Anwendungsfällen sind Tabellenziffern jedoch unansehnlich, weil die Abstände optisch uneinheitlich sind. Deshalb gibt es noch einen Satz an Proportionalziffern.

Für den Fall von URLs und E-Mail-Adressen, die oft aus zusammenhanglosen Zeichenketten bestehen, ist die deutliche Unterscheidbarkeit der Zeichen enorm wichtig. »Linux Libertine« enthält eine gestrichene Null, um die Unterscheidbarkeit von O (Oh) und 0 (Null) in solchen Situationen zu verbessern. Die deutliche Differenzierung der Zeichen I, l, 1 und | (sowie / in der Kursiven) war beim Design der »Linux Libertine« ein wichtiges Kriterium (siehe dazu Abbildung 2).

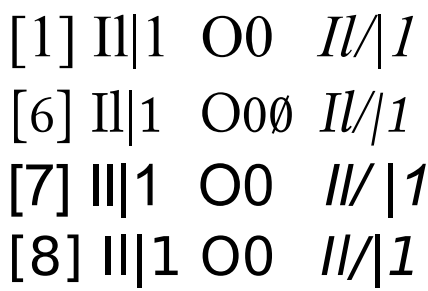


Abbildung 2: Kriterium Differenzierbarkeit. [1] Times New Roman, [6] Linux Libertine, [7] Arial, [8] DejaVu Sans

Abschließend möchte ich noch kurz auf den fünften Ziffernsatz der »Linux Libertine« eingehen – die Römischen Ziffern. Als hübsch werden diese angesehen, wenn die Buchstabenelemente dieser Ziffern verschmelzen wie z. B. bei

„Kaiser Friedrich III." oder „Hartz IV". Die römischen Ziffern der »Linux Libertine« erfüllen nicht nur diesen Habitus, sie weisen darüberhinaus die niedrigere Ziffernhöhe auf (wie auch die lateinischen Normalziffern) und haben das optische Gewicht von Kleinbuchstaben. Während die ersten vier beschriebenen Ziffernsätze mithilfe von OpenType automatisch umgeschaltet werden können, müssen die Römischen Ziffern aufgrund ihrer komplexen Mathematik von Hand ausgewählt werden.

Verwendung mit XeTeX

von *Michael Niedermair*

Auf Basis von Philipp H. Polls künstlerischer Gestaltung konnte ich mich an die Einbindung in XeTeX machen. Da XeTeX von sich aus OpenType-Fonts unterstützt, fiel die Einbindung sehr viel leichter als mit dem alten Paket für die Type-1-Schriften. Das Paket `xelibertine` wird mit dem Makro `usepackage` aufgerufen.

```
\usepackage[<optionen>]{xelibertine}
```

Dabei werden entsprechende Aufrufe des Paketes `fontspec` durchgeführt, welches für die Fonteinbindung in XeTeX zuständig ist.

Optionen

Folgende Optionen sind dabei möglich:

`debug`	Alle Aufrufparameter werden auf der Konsole ausgegeben.
`noamsmath`	Das Laden des Pakets `amsmath` wird nicht durchgeführt. Achtung: Alle Mathematik-Fonts müssen vor dem Paket `xelibertine` geladen werden!
`lucida`	Es wird das Lucida-Font-Paket vor der Schrift »Linux Libertine« geladen (`\usepackage[expert]{lucidabr}`).
`rawfeature`	Es können direkt die *rawfeature* des Pakets `fontspec` genutzt werden. Ein '+' fügt ein Feature hinzu, ein '−' entfernt dieses. Wird beim Aufruf kein Parameter angegeben, so werden die Grundfeatures des Fonts nach der Adobe-Anleitung verwendet.
`language`	Es wird eine Sprache für den Font aktiviert.
`script`	Es wird ein Skript (in Abhängigkeit von der Sprache) für den Font aktiviert.
`dejavusans`	Verwendet den Font *DejaVu Sans* für *sffamily*.
`dejavusansmono`	Verwendet den Font *DejaVu Sans Mono* für *ttfamily*.

`draft`	Es wird der *drafttext* (beispielsweise »Entwurf«) als Hintergrundtext verwendet.
`drafttext`	Der Hintergrundtext.
`noquotes`	Verhindert das Definieren der Anführungszeichen `\glqq` und `\grqq`.

Spezielle Makros

Damit man nicht in die Tiefen der `fontspec`-Dokumentation einsteigen muss, werden einige Makros für das einfachere Arbeiten definiert.

`\libertine`	Schaltet auf den »Linux Libertine«-Font um. Dabei werden die Paketoptionen verwendet. `{\libertine Dies ist ein Text!}`
`\OTF`	Aktiviert *feature*-Tags. Siehe hierzu auch die Option *rawfeature*. `{\OTF{+smcp}Dies ist ein Text!}`
`\Lglyph`	Ein Zeichen kann mit Hilfe des Glyphnamens (siehe Anhang der xelibertine-Dokumentation) gesetzt werden. Ist der Glyphname nicht vorhanden, so wird kein Zeichen gesetzt. Dabei wird auf den »Linux Libertine«-Font umgeschaltet. Verwendet man die `*`-Variante, so wird keine explizite Fontumschaltung durchgeführt. `\Lglyph{Tux}`
`\Leuro`	Es wird das »Linux Libertine«-Euro-Zeichen gesetzt. `\Leuro`
`\Llogo`	Es wird das »Linux Libertine«-Logo gesetzt.`\Llogo`
`\numprp`	Es wird auf die proportionalen Ziffern umgeschaltet. Siehe Abbildung 3.
`\numtab`	Es wird auf die Tabellenziffern umgeschaltet.
`\numold`	Es wird auf die Mediävalziffern-Minuskelziffern umgeschaltet.
`\numzero`	Es wird auf das automatische Ersetzen der normalen Null durch die gestrichene Null umgeschaltet.
`\numfrac`	Es werden Brüche durch ein typografisches Bruchzeichen ersetzt, beispielsweise 1/2.

Auswahl von OpenType-Eigenschaften

Mit den `feature`-Tags werden bestimmte Eigenschaften des Fonts aktiviert. Die Aktivierung der Tags erfolgt mit dem Makro `\OTF`[1]. Der »Linux Libertine«-Font unterstützt folgende Tags:

smcp	(Small Capitals) Minuskeln[2] → Kapitälchen.
c2sc	(Small Capitals From Capitals) Versalien[3] → Kapitälchen.

[1] Alternativ kann auch der `fontspec`-Befehl `\addfontfeature` verwendet werden.
[2] Kleinbuchstaben, auch Gemeine genannt.
[3] Großbuchstaben, auch Majuskel genannt.

liga (Standard Ligatures) Standardligaturen[4], wie z. B. ff, fi, fl, ...
hlig (Historical Ligatures) Historische, heute nicht mehr verwendete Ligaturen: z. B. st und ct.
dlig (Discretionary Ligatures) Nützliche, aber nicht notwendige Ligaturen, wie z. B. tz.
frac (Fractions) Brüche, wie z. B. 1/2 werden durch ein Zeichen ersetzt.
tnum (Tabular Figures) Ziffern für Tabellen.
pnum (Proportional Figures) Proportionale Ziffern (z. B. für Fließtext).
onum (Oldstyle Figures) Mediävalziffern – Minuskelziffern.
zero (Slashed Zero) Automatische Ersetzung der normalen Null durch die gestrichene Null.
salt (Stylistic Alternates) Stilistische Alternativen.
ss01 (Stylistic Set 1) Deutsche Variante der Majuskelumlaute → betonte Vokale (sog. Trema-Buchstaben).
ss02 (Stylistic Set 2) Verwendet teilweise geschwungenere Varianten von Großbuchstaben, bisher nur bei K und R.
ss03 (Stylistic Set 3) Eszetts in SS/ss verwandeln.
fina (Terminal Forms) Besondere Zeichen für das Wortende.
sinf (Scientific Inferiors) Tiefgestellte Zeichen.
sups (Superscript) Hochgestellte Zeichen.
aalt (Access All Alternates) Zeigt alle Alternativen an.

```
{\numprp\ZAHL}

{\numtab\ZAHL}

{\numold\ZAHL}

{\numzero\ZAHL}

{\numfrac 1/2 1/3 2/3 1/4
3/4 1/5 2/5 3/5 4/5 1/6 5/6
1/8 3/8 5/8 7/8}
```

01234567890
01234567890
01234567890
Ø1234567890
½ ⅓ ⅔ ¼ ¾ ⅕ ⅖ ⅗ ⅘ ⅙ ⅚ ⅛ ⅜ ⅝ ⅞

Abbildung 3: Beispiel mit verschiedenen Zahlen

```
\OTF{+dlig}\TEXT
```

„Große Hamburger Straße mit Quelle“
12.345.678,90 €!?
äöüßéà ÄÖÜß fi & ff & ffl & ffi

Abbildung 4: Beispiel mit Ligaturen

[4] Standardmäßig eingeschaltet.

Zusammenfassung

XeTeX hebt lang kritisierte Beschränkungen, wie die Unbenutzbarkeit von OpenType-Fonts endlich auf und ermöglicht eine echte Unicode-Unterstützung sowie die Verwendung aktueller typografischer SmartFont-Technik. Mit der Kombination aus XeTeX und »Linux Libertine« ist damit ein Paket geschaffen, das die Vorteile von LaTeX mit der Notwendigkeit der Internationalisierung verbindet, wobei Belange von Minderheiten ebenso berücksichtigt sind wie die speziellen typografischen Bedürfnisse der Wissenschaften.

Wie am Anfang jeder Entwicklung, gibt es Umstiegshürden (wie in diesem Falle das Fehlen des Pakets `microtype`), die mit steigender Nachfrage bald beseitigt sein werden. Unseren Elchtest hat die Kombination hingegen schon bestanden: Eine 80-seitige Diplomarbeit – bebildert, mit Formeln ausgestattet und allem was sonst dazugehört – wurde perfekt kompiliert.

Links

- Linux Libertine `http://linuxlibertine.sf.net`
- svn: `http://linuxlibertine.svn.sourceforge.net/viewvc/linuxlibertine/trunk/`
- XeTeX-Homepage (nicht immer aktuell, da die XeTeX-Entwicklung inzwischen im TeX Live-Verbund stattfindet) `http://scripts.sil.org/xetex`
- XeTeX-Tutorium (englisch) `http://xml.web.cern.ch/XML/lgc2/xetexmain.pdf`
- fontspec `http://downloads.miek.nl/2008/fontspec.pdf`

Konferenzmanagement mit LaTeX

Uwe Ziegenhagen

Die Organisation von Veranstaltungen wie Konferenzen, Tagungen oder Workshops erfordert eine ganze Reihe von Rechnungen, Teilnehmerlisten, Namensschildern und anderen Dokumenten. LaTeX stellt für alle diese Dokumenttypen geeignete Klassen bereit und über das Paket `datatool` von Nicola Talbot kann man ihnen den Zugriff auf CSV-Dateien (Comma-Separated Values) ermöglichen. In diesem Artikel beschreibe ich anhand eines fiktiven Beispiels die genutzten Pakete und ihr Zusammenwirken.

Einleitung

Ausgangspunkt der Betrachtungen sei eine potentielle Tagung, für deren Organisation die folgenden Dokumente benötigt werden:

- Aktuelle Teilnehmerlisten mit dem Zahlungsstatus der Konferenzgebühr;
- ein Tagungsband, der die einzelnen Beiträge der Teilnehmer enthält;
- Rechnungen und Teilnahmebestätigungen;
- Namensschilder.

Die Grundlage für die Erstellung nahezu all dieser Dokumente bildet eine Excel- bzw. OpenOffice-Tabelle, die dann über den Umweg einer CSV-Datei von LaTeX verarbeitet wird.

Das Paket `datatool`

Das Paket `datatool` [2] von Nicola Talbot, das ihr Paket `cvstools` [1] ablöst, stellt eine einfache Schnittstelle zur Verarbeitung von CSV-Dateien bereit. Das Paket besteht eigentlich aus mehreren Style-Dateien (siehe nachfolgende Tabelle). In diesem Artikel liegt der Fokus jedoch auf der Datei `datatool.sty`; für die anderen Teile sei auf die sehr gute Dokumentation des Pakets verwiesen.

Für die weitere Betrachtung wird eine Datenbankdatei von Teilnehmenden entsprechend Listing 1 angenommen.

Tabelle 1: Bestandteile des Pakets `datatool`

`datatool.sty`	erstellt und exportiert Daten-Dateien, importiert extern erstellte Daten-Dateien
`datapie.sty`	erstellt Kuchen-Diagramme (pie charts)
`dataplot.sty`	generiert zweidimensionale Punkt- oder Liniendiagramme
`databar.sty`	erstellt Balkendiagramme
`databib.sty`	konvertiert BibTeX-Bibliographien in Daten-Dateien

Listing 1: Datenbankdatei `datad.csv`

```
vorname,nachname,strasse,ort,zuzahlen,bezahlt
Nicole,Möller,Schillerplatz 61,18419 Vogelow,100.0,100.0
Tom,Lehmann,Nachtigallgasse 11,29098 Altaue,100.0,100.0
Tim,Wagner,Amselplatz 92,46917 Langenhausen,100.0,0.0
Moritz,Müller,Waldallee 71,55348 Kirchstein,100.0,0.0
Susi,Mayer,Sonnenweg 27a,83675 Heidehausen,100.0,100.0
Ines,Mayer,Wasserallee 83a,26118 Kirchfurt,100.0,100.0
Uwe,Meier,Sonnenplatz 7,07514 Vogelburg,100.0,0.0
Mandy,Berger,Goetheweg 25,03783 Wolfental,100.0,100.0
Tim,Grünwald,Wiesenplatz 9a,90778 Moosow,100.0,50.0
Jenny,Köster,Finkenallee 29c,53522 Wiesenow,100.0,100.0
Marko,Mayer,Amselweg 11c,32108 Grünstein,100.0,100.0
Jenny,Berger,Wiesenallee 82,72044 Moosaue,0.0,0.0
```

Listing 2 zeigt ein minimales Beispiel, das die Vor- und Nachnamen der Teilnehmer in einer Liste (siehe Tabelle 1) ausgibt. Die Daten werden aus der Datei `datad.csv` entnommen (siehe Listing 1).

Mit dem Befehl `\DTLloaddb{list}{datad.csv}` werden der Name der Datenquelle und die Quelldatei der Datenbank festgelegt, hier in unserem fiktiven Beispiel `list` als Name für die Datei `datad.csv`.

`\DTLforeach{arg1}{arg2}{arg3}` definiert, was mit den aus der Datenbank gelesenen Datensätzen geschehen soll. `arg1` spezifiziert den Namen der Datenquelle, `arg2` nimmt die Zuordnung der Spalten zu den entsprechenden LaTeX-Befehlen vor und `arg3` legt die Ausgabe fest.

Listing 2: `listed.tex` – Quellcode für Tabelle 1

```
\documentclass{article}
\usepackage{datatool}  \usepackage[latin1]{inputenc}
\begin{document}
```

```
\DTLloaddb{list}{datad.csv}

\begin{tabular}{ll}
\bfseries Vorname & \bfseries Nachname
\DTLforeach{list}{\first=vorname,\last=nachname}{%
\\ \first & \last }
\end{tabular}
\end{document}
```

Vorname	**Nachname**
Nicole	Möller
Tom	Lehmann
Tim	Wagner
Moritz	Müller
Susi	Mayer
Ines	Mayer

Abbildung 1: `listed.pdf` – Ausgabe von Listing 2 (Auszug)

Listing 3 enthält die notwendigen Befehle, um alle Spalten der Datenbank auszuwerten und führt zwei neue Befehle ein: `\DTLsort` und `\DTLsumforkeys`. `\DTLsort{key}{database}` sortiert die Einträge der Datenbank anhand des Parameters `key`. Es können auch mehrere Werte an `key` übergeben werden. Dies ist sinnvoll, wenn gleiche Nachnamen in der Datenbank auftreten und deshalb zusätzlich nach dem Vornamen sortiert werden soll. Die Version des Befehls ohne Stern `*` berücksichtigt Groß- und Kleinschreibung (Großbuchstaben vor Kleinbuchstaben), die Version mit Stern ignoriert diese beim Vergleich von Zeichenketten. Der Befehl akzeptiert als optionales Argument eine Liste von »Schlüsseln«, auf die zurückgegriffen wird, wenn ein Eintrag unzureichend ist.

`\DTLsumforkeys{database}{key}{\command}` summiert die Werte aller `key`-Felder und speichert diese im Befehl `\command`. Mit diesen beiden Befehlen können wir jetzt die Teilnehmerliste nach dem Nachnamen sortieren und die erwarteten und eingegangenen Zahlungen zusammenzählen (vgl. Tabelle 3 und das Listing 3).

Listing 3: `liste2d.tex` – Quellcode für Tabelle 2

```
\documentclass{article}
\usepackage{datatool}
\usepackage[latin1]{inputenc}
```

```
\usepackage{eurosym}
\pagestyle{empty}
\begin{document}

\DTLloaddb{list}{datad.csv}
\DTLsort{nachname}{list}
\DTLsumforkeys{list}{zuzahlen}{\soll}
\DTLsumforkeys{list}{bezahlt}{\haben}

\begin{tabular}{rllrr}
Nr. & \bfseries Vorname & \bfseries Nachname &
      \bfseries Gebühr &  \bfseries Zahlung
\DTLforeach{list}{\first=vorname,\last=nachname,
                  \fee=zuzahlen, \paid=bezahlt}{%
  \DTLiffirstrow{\\ \hline}{\\}%
  \theDTLrowi & \first & \last & \fee\,\euro & \paid\,\euro
}
\end{tabular}

\soll\,\euro{} sind zu bezahlen, \haben\,\euro{} sind bezahlt.
\end{document}
```

Um besser zwischen einzelnen Zeilen unterscheiden zu können, können u. a. die Tabellenzeilen mit dem Paket `colortbl` eingefärbt werden, und zusammen mit dem Befehl `\DTLifoddrow` verbunden werden. Listing 4 zeigt den Quellcode, der für Tabelle 3 benötigt wird.

Listing 4: `liste2colord.tex` – Quellcode für Tabelle 3

```
\documentclass{article}
\usepackage[latin1]{inputenc}
\usepackage{datatool,colortbl,xcolor,eurosym}
\begin{document}

\DTLloaddb{list}{datad.csv}
\DTLsort{nachname}{list}
\DTLsumforkeys{list}{zuzahlen}{\soll}
\DTLsumforkeys{list}{bezahlt}{\haben}

\begin{tabular}{rllrr}
\bfseries ID & \bfseries Vorname & \bfseries Nachname & \bfseries Gebühr↩
   & \bfseries Zahlung \\ \hline
\DTLforeach{list}{\first=vorname,\last=nachname, \fee=zuzahlen,\paid=↩
  bezahlt}{%
```

```
\\\DTLifoddrow{\rowcolor{cyan}}{\rowcolor{lime}} %
\theDTLrowi & \first & \last & \fee & \paid  }
\end{tabular}

\soll\,\euro{} sind zu bezahlen, \haben\,\euro{} sind bezahlt.
\end{document}
```

Nr.	**Vorname**	**Nachname**	**Gebühr**	**Zahlung**
1	Mandy	Berger	100.0 €	100.0 €
2	Jenny	Berger	0.0 €	0.0 €
3	Tim	Grünwald	100.0 €	50.0 €
4	Jenny	Köster	100.0 €	100.0 €
5	Tom	Lehmann	100.0 €	100.0 €
6	Susi	Mayer	100.0 €	100.0 €
7	Ines	Mayer	100.0 €	100.0 €
8	Marko	Mayer	100.0 €	100.0 €
9	Uwe	Meier	100.0 €	0.0 €
10	Nicole	Möller	100.0 €	100.0 €
11	Moritz	Müller	100.0 €	0.0 €
12	Tim	Wagner	100.0 €	0.0 €

1,100 € sind zu bezahlen, 750 € sind bezahlt.

Abbildung 2: Ausgabe von Listing 3

ID	**Vorname**	**Nachname**	**Gebühr**	**Zahlung**
1	Mandy	Berger	100.0	100.0
2	Jenny	Berger	0.0	0.0
3	Tim	Grünwald	100.0	50.0
4	Jenny	Köster	100.0	100.0
5	Tom	Lehmann	100.0	100.0
6	Susi	Mayer	100.0	100.0
7	Ines	Mayer	100.0	100.0
8	Marko	Mayer	100.0	100.0
9	Uwe	Meier	100.0	0.0
10	Nicole	Möller	100.0	100.0
11	Moritz	Müller	100.0	0.0
12	Tim	Wagner	100.0	0.0

1,100 € sind zu bezahlen, 750 € sind bezahlt.

Abbildung 3: Ausgabe von Listing 4

Der LaTeX-Code wird etwas unübersichtlicher, wenn wir die CSV-Dateien nicht mit Komma-Separatoren und Dezimalpunkten nutzen, weil wir die deutschen Einstellungen (Komma als Dezimaltrenner, Semikolon als Spaltentrenner) berücksichtigen wollen oder müssen.

Listing 5 zeigt die aus einer deutschen Version von Excel exportierte CSV-Datei, Listing 6 den LaTeX-Code und Tabelle 4 die generierte Ausgabe. Listing 6 führt drei neue Befehle ein, die einer genaueren Erklärung bedürfen:

- `\DTLifcurrency{text}{truepart}{falsepart}`: Wenn `text` eine Währung ist, dann nutze `truepart`, ansonsten `falsepart`.
- `\DTLconverttodecimal{number}{command}` konvertiert eine Zahl aus einem lokalen Format in das Dezimalformat und speichert das Ergebnis in `command`.
- `\DTLdecimaltocurrency{number}{command}` konvertiert eine Dezimalzahl in das lokale Format.

Listing 5: Die Datenquelle `data2d.csv` mit deutscher Formatierung

```
vorname;nachname;strasse;ort;zuzahlen;bezahlt
Nicole;Möller;Schillerplatz 61;18419 Vogelow;100,0;100,0
Tom;Lehmann;Nachtigallgasse 11;29098 Altaue;100,0;100,0,0
Tim;Grünwald;Wiesenplatz 9a;90778 Moosow;100,0;50,0
Jenny;Köster;Finkenallee 29c;53522 Wiesenow;100,0;100,0
Marko;Mayer;Amselweg 11c;32108 Grünstein;100,0;100,0
Jenny;Berger;Wiesenallee 82;72044 Moosaue;0,0;0,0
```

Listing 6: `OverviewXLSd.tex`, Quelltext für Tabelle 4

```
\documentclass[10pt]{scrartcl}
\usepackage[latin1]{inputenc}
\usepackage[landscape,a4paper]{geometry}
\usepackage{datatool,eurosym}
\usepackage[ngerman]{babel}
\begin{document}

\section*{Teilnehmer, Stand \today}
\DTLsetseparator{;}      \DTLsetnumberchars{}{,}
\DTLsetdefaultcurrency{\texteuro~}
\DTLloaddb{list}{data2d.csv}
\DTLsort{nachname,vorname}{list}
\DTLsumforkeys{list}{zuzahlen}{\soll}
\DTLsumforkeys{list}{bezahlt}{\haben}
```

```
\begin{tabular}{rllllrr}
ID & Vorname & Nachname & Stra"se & Ort & zu zahlen & bezahlt
\DTLforeach{list}{% definiere Listenelemente
 \first=vorname,\last=nachname,\address=strasse,
 \town=ort,\fee=zuzahlen,\paid=bezahlt}{%
  \DTLiffirstrow{\\ \hline}{\\}%
  \theDTLrowi & \first & \last & \address & \town & \euro\, %
  \DTLifcurrency {\fee}{% wenn \fee eine Waehrungsangabe ist
    \DTLconverttodecimal{\fee}{\fee} % dann konvertiere nach decimal
          % und speicher den Wert wieder in \fee
    \fee  % gebe den Wert von \fee aus
    }{\fee}% wenn keine Waehrung, dann nur ausgeben
  & \euro\, %
  \DTLifcurrency{\paid}{% wenn \paid eine Waehrungsangabe ist
    \DTLconverttodecimal{\paid}{\paid}\paid % dann konvertiere
  }{\paid}% sonst Ausgabe von \paid
} \\ \hline
& & & & \DTLdecimaltocurrency{\soll}{\soll}\euro\,\soll& %
\DTLdecimaltocurrency{\haben}{\haben}\euro\,\haben \\ \hline \hline
\end{tabular}
\end{document}
```

Teilnehmer, Stand 25. Januar 2009

ID	Vorname	Nachname	Straße	Ort	zu zahlen	bezahlt
1	Jenny	Berger	Wiesenallee 82	72044 Moosaue	€ 0,0	€ 0,0
2	Mandy	Berger	Goetheweg 25	03783 Wolfental	€ 100,0	€ 100,0
3	Tim	Grünwald	Wiesenplatz 9a	90778 Moosow	€ 100,0	€ 50,0
4	Jenny	Köster	Finkenallee 29c	53522 Wiesenow	€ 100,0	€ 100,0
5	Tom	Lehmann	Nachtigallgasse 11	29098 Altaue	€ 100,0	€ 100,0
6	Ines	Mayer	Wasserallee 83a	26118 Kirchfurt	€ 100,0	€ 100,0
7	Marko	Mayer	Amselweg 11c	32108 Grünstein	€ 100,0	€ 100,0
8	Susi	Mayer	Sonnenweg 27a	83675 Heidehausen	€ 100,0	€ 100,0
9	Uwe	Meier	Sonnenplatz 7	07514 Vogelburg	€ 100,0	€ 0,0
10	Nicole	Möller	Schillerplatz 61	18419 Vogelow	€ 100,0	€ 100,0
11	Moritz	Müller	Waldallee 71	55348 Kirchstein	€ 100,0	€ 0,0
12	Tim	Wagner	Amselplatz 92	46917 Langenhausen	€ 100,0	€ 0,0
					€ 1100,00	€ 750,00

Abbildung 4: Ausgabe von Listing 6

Briefe und Rechnungen

Für formelle Briefe und Rechnungen gibt es u. a. die Klassen `g-brief` und `scrlttr2`, mit deren Hilfe sich den deutschen Formvorschriften entsprechende Dokumente erstellen lassen. Ich nutze zum einen `scrlttr2`, da sich mit dieser Dokumentenklasse auch komplexe Layouts gut umsetzen lassen, sowie ein selbst entwickeltes Paket, `varsfromjobname.sty`. Mit diesem Paket [3] ist es möglich, einzelne Elemente aus dem Namen der LaTeX-Datei zu extrahieren und in Variablen abzulegen, die dann im Dokument gesetzt werden können. Dazu erwartet das Paket den Dateinamen in der Form `arg1-arg2-....tex`. Insgesamt werden bis zu neun Argumente unterstützt, die dann über `\getfromjobname{argx}` bzw. über die Befehle `\getonefromjobname` bis `\getninefromjobname` zur Verfügung gestellt werden. Auf diese Weise ist es zum Beispiel möglich, Angaben wie Rechnungsdatum und Rechnungsnummer fest im Dateinamen einzubetten.

Für die Erstellung von Serienbriefen gibt es auch mehrere Möglichkeiten. KOMA-Script verfügt über die Möglichkeit, mit Adressdateien umzugehen. Allerdings müssen diese in einem bestimmten Format vorhanden sein, das sich vom für `datatool` genutzten CSV-Format unterscheidet.

Um nur eine Datenquelle einpflegen zu müssen, nutze ich daher auch das `datatool` Paket. Listing 7 zeigt ein Beispiel für die Erstellung eines Serienbriefes mit `datatool`. Die `letter`-Umgebung wird dabei in den Schleifenteil des `\DTLforeach` gesetzt, der Adressparameter für den Brief wird dann mit `{\first \last \address \\\town}` gefüllt.

Listing 7: `scrlttsample.tex`, Quelltext für Abbildung 5

```
\documentclass[a5paper]{scrlttr2}
\usepackage[english]{babel}
\usepackage[latin1]{inputenc}
\usepackage{datatool}
\setkomavar{title}{Teilnahmebestätigung}

\begin{document}
\DTLloaddb{list}{datad.csv}
\DTLforeach{list}{\first=vorname,\last=nachname,\address=strasse,
\town=ort,\fee=zuzahlen,\paid=bezahlt}{% Beginn Schleife

\begin{letter}{\first~\last \\ \address \\ \town}%
\opening{Sehr geehrte/r \first~\last,}%
```

```
Wir bestätigen Ihre Teilnahme am Foobar
Workshop in Musterstadt.

\setkomavar{fromname}{Das Organisationsteam}%
\closing{Mit freundlichen Grüßen}%
\end{letter}
}% Ende Schleife
\end{document}
```

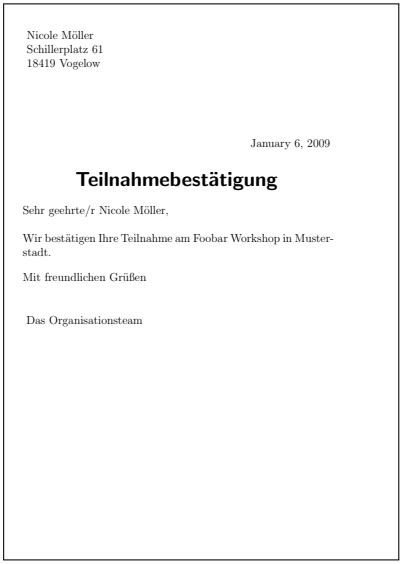


Abbildung 5: Ausgabe von Listing 7

Das Erläuterte ist nur ein einfaches Beispiel, mit den zahlreichen weiteren `datatool`-Befehlen sind noch viel detailliertere Briefe möglich. So lässt sich beispielsweise über ein zusätzliches Feld `Geschlecht` in der Datendatei, das mit »m« oder »w« gefüllt ist, und dem Kommando `\DTLifstringeq` eine geschlechtsspezifische Anrede erstellen.

Namensschilder mit `ticket.sty`

Mit `ticket.sty` lassen sich Namensschilder, Adressaufkleber und ähnliche Dokumente einfach setzen. Listing 8 zeigt den Aufbau einer entsprechenden Datei.

Nach dem Laden diverser Pakete wird das standardmäßig definierte Ticket zurückgesetzt, damit die eigene Definition erfolgen kann. Die Kommandos zwischen `\makeatletter` und `\makeatother` setzen oder unterdrücken Rahmen, Schnitt- und Falzmarken. Der Parameter `\@boxedtrue` ist während der Entwurfsphase der Etiketten sehr nützlich, vor dem Druck sollte er auf `\@boxedfalse` gesetzt werden.

Der Parameter `badges` verweist beim Laden von `ticket.sty` auf die Datei `badges.tdf` (siehe Listing 9), in der die Anzahl und Größe der einzelnen Schilder sowie Angaben über den druckerspezifischen horizontalen und vertikalen Offset stehen, also den Abstand des ersten Tickets vom oberen und linken Rand. Die Angabe des Offsets ist notwendig, da die Ränder über das Paket `geometry` auf Null gesetzt sind.

Der Befehl `\mylabel` wird anschließend definiert, um eine bequeme Schnittstelle zu den einzelnen Namensschildern zu haben, die nur noch Parameter für Name und Ort benötigt.

Listing 8: Quellcode für Abbildung 6

```
\documentclass[a4paper,12pt]{letter}
\usepackage[total={210mm,297mm},top=0mm, %
          left=0mm, includefoot]{geometry}
\usepackage[badges]{ticket}
\usepackage{graphicx,palatino}
\usepackage[latin1]{inputenc}
\usepackage{xcolor}

\renewcommand{\ticketdefault}{}%
\makeatletter
\@boxedfalse % Rahmen um Ticket
\@emptycrossmarkfalse % Falzmarken
\@cutmarktrue % Schnittmarken
\makeatother

\newcommand{\mylabel}[2]{\ticket{%
  \put(7,35){\scalebox{2}{\textbf{#1}}}
  \put(7,25){\scalebox{1.5}{\textbf{#2}}}
```

```
  \put(7,5){\scalebox{1}{\textcolor{gray}{%
    \textit{\LaTeX{} Convention 2009}}}}
}}

\begin{document}
\mylabel{Max Mustermann}{Berlin}
\mylabel{Maria Mustermann}{Berlin}
\mylabel{Marian Mustermann}{Berlin}
\mylabel{Micky Mustermann}{Berlin}
\mylabel{Mario Mustermann}{Berlin}
\mylabel{Markus Mustermann}{Berlin}
\end{document}
```

Listing 9: `badges.tdf` – Definition des Papierbogens `badges.tdf`

```
\unitlength=1mm
\hoffset=-10mm
\voffset=-16mm
\ticketNumbers{2}{5}
\ticketSize{90}{55.2} % unitlength => mm
\ticketDistance{0}{0} % unitlength => mm
```

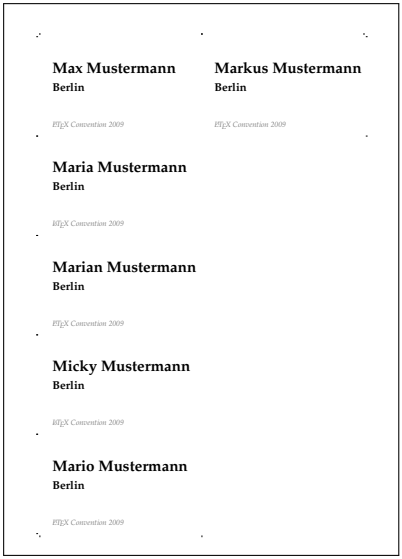


Abbildung 6: Ausgabe von Listing 8

Natürlich ist auch hier das Ziel, die Namensschilder durch `datatool` automatisch setzen zu lassen, Listing 10 zeigt die angepasste Datei mit den dafür notwendigen `datatool`-Befehlen.

Listing 10: `badgesd.tex` – Quellcode für Abbildung 7

```
\documentclass[a4paper,12pt]{letter}
\usepackage[total={210mm,297mm},top=0mm,
        left=0mm, includefoot]{geometry}
\usepackage[badges]{ticket}
\usepackage{graphicx,palatino,marvosym}
\usepackage[latin1]{inputenc}
\usepackage{xcolor}
\usepackage{datatool}

\renewcommand{\ticketdefault}{}%
\makeatletter
  \@emptycrossmarkfalse
  \@cutmarkfalse
  \@boxedtrue
\makeatother

\newcommand{\mylabel}[2]{\ticket{%
  \put(7,35){\scalebox{2}{\textbf{#1}}}
  \put(7,25){\scalebox{1.5}{\textbf{#2}}}
  \put(7,5){\scalebox{1}{\textcolor{gray}{%
      \textit{\LaTeX{} Convention 2009}}}}
}}

\begin{document}

\DTLloaddb{list}{datad.csv}
\DTLforeach{list}{\first=vorname,\last=nachname,
\ort=ort,\fee=zuzahlen,\paid=bezahlt}
{%
\mylabel{\first\ \last}{\ort}
}

\end{document}
```

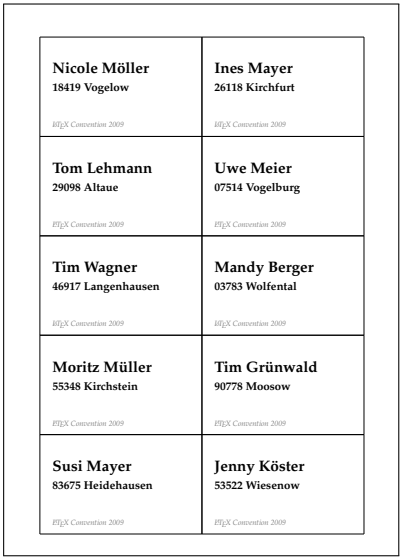


Abbildung 7: Ausgabe von Listing 10

Dokumente zusammenführen mit `combine.sty`

Bei größeren Workshops oder Tagungen ist es oft üblich, dass die Beiträge der Teilnehmer in Tagungsbänden, den Proceedings, veröffentlicht werden. Die Hauptarbeit der Editoren besteht dann darin, die einzelnen Beiträge zu einem kohärenten Dokument zusammenzufügen, was sehr aufwändig sein kann. Als Alternative zu dieser manuellen Integration lässt sich das Paket `combine.sty` nutzen, das eigenständig kompilierbare Dokumente zu einem Text »zusammenbaut«.

Listing 11 zeigt einen minimalen Beitrag eines Teilnehmers, der – zusammen mit gleichartigen Dateien – zu einem einzigen Dokument zusammengesetzt werden soll.

Eine einfache `combine`-Datei, die zwei dieser Einzelbeiträge in eine Datei kompiliert und ein Inhaltsverzeichnis ablegt, ist in Listing 12 abgedruckt. Lokal in den Dateien geladene Pakete werden in dieses Hauptdokument übernommen; daher empfiehlt es sich, eine für die Teilnehmenden verpflichtende Vorlage zu erstellen und keine weiteren Pakete zuzulassen.

Listing 11: `import1.tex` – Quellcode eines zu importierenden Dokuments

```
\documentclass{article}

\author{Max Mustermann}
\title{Anmerkungen zum Euro}

\begin{document}
\maketitle

Text für den Artikel.

\end{document}
```

Listing 12: `combineit.tex` – Quellcode eines `combine` Dokuments

```
\documentclass{combine}
\pagestyle{combine}
\begin{document}

\tableofcontents
\begin{papers}
  \coltoctitle{Anmerkungen zum Euro} % Erster Titel für ToC
  \coltocauthor{Max Mustermann} % Erster Autor für ToC
  \label{import1}
  \import{import1}

  \coltoctitle{Anmerkungen zur D-Mark} % Zweiter Titel für ToC
  \coltocauthor{Maria Mustermann} % Zweiter Autor für ToC
  \label{import2}
  \import{import2}
\end{papers}
\end{document}
```

Zusammenfassung

Von den Möglichkeiten, die LaTeX für den Satz von dynamischen Dokumenten bietet, konnte dieser Artikel nur einige wenige zeigen. Insbesondere der Satz von XML-basierten Dokumenten oder von Inhalten, die nur per Datenbanksystem zugänglich sind, bietet spannende Projekte für den Satz mit LaTeX und wird eventuell noch in einer der nächsten Ausgaben von »Die TeXnische Komödie« nachgereicht.

Für Kommentare, Hinweise und Anmerkungen bin ich jederzeit dankbar; am einfachsten erreicht man mich über das Kontaktformular meiner Homepage `http://www.uweziegenhagen.de`. Dort werden auch die genutzten Dateien und Errata für diesen Artikel bereitgestellt.

Literatur

[1] Nicola Talbot: *CSV Tools*; 2007; `CTAN:tex-archive/macros/latex/contrib/csvtools`.

[2] Nicola Talbot: *datatool v 1.01: Databases and data manipulation*; 2007; `CTAN:tex-archive/macros/latex/contrib/datatool`.

[3] Uwe Ziegenhagen: *varsfromjobname v 0.5*; 2009; `CTAN:tex-archive/macros/latex/contrib/varsfromjobnames`.

Magazinerstellung mit LaTeX

Dominik Wagenführ

Mit LaTeX lassen sich viele Dokumente setzen. Allen voran natürlich wissenschaftliche Arbeiten, aber auch Briefe oder Präsentationen sind damit kein Problem. Dass sich auch ein PDF-Magazin damit erstellen lässt, soll dieser Artikel zeigen und vor allem, wie man einige auftretende Probleme lösen kann.

Einleitung

freiesMagazin [2] ist – wie der Name sagt – ein freies Magazin, welches sich monatlich mit Themen rund um Linux und Open Source beschäftigt. Die erste Ausgabe erschien im März 2006 noch als Newsletter in OpenOffice.org. Obwohl das Layout ziemlich einfach gehalten war und auch nur wenige Bilder eingebunden wurden, gab es sehr häufig Probleme mit dem Satz. Vor allem die Platzierung der Bilder und der korrekte Umbruch führten ab und an zum Fluchen. Ein weiterer Nachteil des damaligen OpenOffice-Formates: In heutigen Versionen der Büro-Suite sind die Seiten falsch umgebrochen und die Bilder nicht korrekt gesetzt. Eine »Rekompilierung« der Ausgabe von damals ist also nicht ohne weitere Arbeit möglich.

Aus diesem Grund hat die Redakteurin Eva Drud auf das Textsatzsystem LaTeX umgestellt, welches ihr bereits von der Diplomarbeit bekannt war. So konnte sie sich mehr auf den Inhalt des Magazins konzentrieren und der Textsatz wurde von LaTeX übernommen. Des Weiteren wurde aufgrund der Stabilität, die aus der sehr langen Entwicklungszeit resultiert, und natürlich wegen der Offenheit für LaTeX entschieden, denn **freies**Magazin setzt bei der Erstellung wie auch bei den Inhalten auf freie Software.

Entwicklung

Für die erste LaTeX-Ausgabe von **freies**Magazin wurde auf das Beispieldokument von Bob Kerstetter [4] zurückgegriffen. Die einzelnen Artikel wurden zwei- oder dreispaltig mithilfe des Pakets multicols gesetzt. Im Laufe der Monate hat sich vor allem am Design des Magazins einiges geändert, die Basis – der

LaTeX-Code – aber blieb größtenteils identisch. Es gab ein Hauptdokument, welches alle LaTeX-Pakete lud, das Layout einstellte und dann alle Unterseiten per `\include` einband. Dies führte unter anderem zu einer hohen Redundanz. So mussten für alle Artikel, obwohl deren Kopf stets gleich aufgebaut war, immer die Angaben

```
\begin{multicols}{2}
[\textbf{{\Large{\hypertarget{MARKE}{TITEL}\label{MARKE}}} \hspace*{4mm}↩
  von AUTOR\\}
\color{orange}
\leftline{\rule{.33\linewidth}{2pt}}
\color{black}]
```

manuell eingefügt werden. Auch die Angaben im Inhaltsverzeichnis waren nicht sonderlich übersichtlich (als Teil einer Tabelle):

```
\hyperlink{MARKE}{TITEL} && \hyperlink{MARKE}{\textbf{S.~\pageref*{MARKE↩
  }}}\\
```

Auf diese Art wurde weitergemacht, bis im Mai 2007 die Übersichtlichkeit in den einzelnen Artikeln immer mehr verloren ging. Vor lauter LaTeX-Befehlen und -Formatierungen sah man den eigentlichen Text kaum noch. Daher wurde begonnen, für immer wiederkehrende Befehle neue Kommandos zu definieren, anfangs nur für Bilder. So konnte man mit dem folgenden Code leicht ein Bild im Text einbinden und man musste nicht immer den gesamten Code-Block samt `figure`- und `minipage`-Umgebung kopieren. Neben der Code-Optimierung und Übersichtlichkeit hatten diese Befehlsdefinitionen aber weitere Vorteile: Die Artikel wurde zum einen gleichförmiger und leichter erweiterbar. Passierte es vorher immer einmal, dass man beim Kopieren vielleicht eine Zeile oder eine Zahl vergaß, kann das nun nicht mehr so leicht passieren. Zum anderen war es davor nur mit viel Aufwand möglich, beispielsweise alle Bildunterschriften kursiv zu formatieren. Nach der obigen Änderung muss man nur eine Zeile anpassen und das neue Format wird auf alle Bilder angewandt.

```
\bild{BILD}{UNTERTITEL}{BREITE}
```

Das Konzept der Makros wurde auf immer mehr Code-Blöcke angewandt, so auch der Beginn eines Artikels oben, der dann nur noch aus dem Aufruf

```
\Artikelstart{TITEL}{AUTOR}{MARKE}
```

bestand. Der Nachteil war, dass im September 2007 die Hauptdatei zum Großteil aus Makrodefinitionen bestand, an denen der Redakteur nicht sonderlich interessiert war. Für die Oktoberausgabe wurde die Hauptdatei aufgespalten

und drei neue Dateien (`pakete.tex`, `befehle.tex` und `layout.tex`) erstellt, die sich jeweils nur mit ihrem Gebiet beschäftigen. Dieses Vorgehen wurde bis heute beibehalten; die drei Dateien, die die Basis des Magazins bilden, stehen jeden Monat auf der Homepage von **freies**Magazin [1] zum D ownload bereit und können unter der GNU Free Documentation License (GFDL) [3] weiterverwendet werden.

Die Befehlsdefinitionen wurden im Laufe des Jahres bis zum Oktober 2008 immer mehr erweitert und verbessert, sodass auch einige neue LaTeX-Umgebungen hinzu gekommen sind. Die anfangs 200 Zeilen der Der Datei `befehle.tex` sind inzwischen auf fast 800 angewachsen, wobei die Dokumentation der Datei aber sicherlich fünfzig Prozent davon einnimmt.

Im Folgenden sollen einige Probleme der Magazinerstellung und deren (hoffentlich sinnvolle) LaTeX-Lösung genauer betrachtet werden. Es sei aber dazu gesagt, dass einige Umsetzungen nicht ganz sauber sind und z. B. einige Warnungen bei der Kompilierung mit `pdflatex` erzeugen. Es wurde in diesen Fällen bisher keine bessere Lösung gefunden.

Probleme und Lösungen

Textumflossene Bilder

Eines der größeren Probleme in LaTeX sind textumflossene Bilder. Hierzu eignet sich meiner Meinung nach das Paket wrapfig [6] am besten. Über den folgenden Code kann man ein textumflossenes Gebiet ausschneiden, wobei `INHALT` in dem Fall das Bild wäre, `ZEILEN` gibt die Anzahl der Zeilen und `BREITE` die Breite der Aussparung an und die `AUSRICHTUNG` kann mit »l« (links) oder »r« (rechts) bestimmt werden.

```
\begin{wrapfigure}[ZEILEN]{AUSRICHTUNG}{BREITE}
INHALT
\end{wrapfigure}
```

Etwas komplizierter sind textumflossene Grafiken bei mehrspaltigem Satz. Es geht meines Wissens bis heute nicht vollautomatisch, aber mithilfe der Anleitung auf der Webseite [7] konnten immerhin verschiedene Befehle definiert werden, die bei der Ausrichtung helfen.

Zuerst die wichtigen Befehlsdefinitionen:

```
% stellt das Bild dar
% (Standardgroesse: eine Spaltenbreite)
\newcommand{\Bild}[3][1]
```

```
{{\centering
\begin{minipage}[t]{#1\linewidth}
  \centering
  \includegraphics[width=\textwidth]{#2}\\
  \emph{#3}
\end{minipage}}}

% erzeugt einen eingerueckten Absatz
\newcommand{\Bildabsatz}[3]
{\begin{wrapfigure}[#1]{#2}{#3}
\vfill
\end{wrapfigure}}

% erzeugt einen kompletten Absatz
\newcommand{\BildabsatzVoll}[1]
{\linebreak[4] \vspace*{#1}

}

% erzeugt einen eingerueckten Absatz mit Bild
\newcommand{\BildabsatzMitBild}[4]
{\begin{wrapfigure}[#1]{#2}{#3}
#4
\end{wrapfigure}}
```

Möchte man ein Bild beispielsweise zweispaltig einfügen, benutzt man

```
\Bild[2.0]{bild.png}{Untertitel.}
```

Auf diese Art ragt das Bild aber in den Text der zweiten Spalte hinein. Daher sucht man sich in dieser zweiten Spalte die Zeile aus, an der man umbrechen will und fügt direkt im Text (also ohne Leerzeilen davor und danach)

```
\BildabsatzVoll{5cm}
```

ein, wobei die Höhe natürlich je nach Bild angepasst werden muss. Der Text wird an der Stelle einfach um 5 cm nach unten geschoben. Sollte das Bild am unteren Bildrand ausgerichtet sein, empfiehlt sich ein

```
\BildabsatzVoll{1cm} \columnbreak
```

Dies lässt etwas Platz und bricht dann in die nächste Spalte um. Den Umbruch nur über `\vspace` zu regeln, ist mir nicht möglich gewesen. In vielen Fällen

sehen textumflossene Grafiken aber schöner aus. Will man z. B. ein Bild über anderthalb Spalten einfügen, benutzt man

```
\Bild[1.5]{bild.png}{Untertitel.}
```

Dies sucht sich erneut die Zeile in der zweiten Spalte, an der man umbrechen will und fügt

```
\Bildabsatz{10}{l}{4cm}
```

ein, wobei die Anzahl der Zeilen (10) und die Breite der Spalte (4 cm) vom Bild bzw. Layout abhängen. Das Ganze geht natürlich auch so, dass das Bild links umflossen wird. Es wird dann an der gewünschten Stelle mittels

```
\BildabsatzMitBild{10}{r}{4cm}
{\Bild[3.1]{bild.png}{Untertitel.}}
```

eingebunden. Anstelle des `\vfill` beim `\Bildabsatz` wird also das Bild angezeigt (Höhe und Breite muss man natürlich wieder selbst anpassen).

Trickreich ist die Angabe der Bildgröße. Diese richtet sich nämlich immer nach der Umgebung, in der sie eingebettet ist, welche nun `\wrapfigure` mit halber Spaltenbreite ist. Daher muss das Bild in etwa um den Faktor 3.1 der aktuellen Breite angezeigt werden, damit es über anderhalb Spalten geht. In der zweiten Spalte muss wie oben per `\BildabsatzVoll` noch ein Leerraum gelassen werden.

Link- und Quellzähler

Für ein PDF-Magazin, welches im Internet erscheint, sind Links unumgänglich. Diese kann man natürlich per Hand nummerieren. Will man jedoch einmal einen Absatz nach vorne setzen, müsste man gegebenenfalls sehr viele Zeilen ändern. Aus diesem Grund wurde für die Links und die Quellen in **freies**Magazin eine automatische Zählweise eingeführt:

```
% neue Zaehler
\newcounter{linkcounter}
\newcounter{quellcounter}

% Link im Text
\newcommand{\Link}[2]
[\arabic{linkcounter}\refstepcounter{linkcounter}]
{\href{#2}{[#1]}}

% Quelle
```

```
\newcommand{\Quelle}[1]
{[\arabic{quellcounter}]\refstepcounter{quellcounter} & \url{#1}}\\
```

Es gibt zwei Zähler: `linkcounter` und `quellcounter`. Der Zähler `linkcounter` zählt dabei die Links im Text, die über

```
\Link{http://www.zieladresse.de}
```

angegeben werden. Es wird dabei aber immer nur die Zahl `[X]` verlinkt. Die Quellen werden am Ende eines Artikels aufgelistet. Es war mir dabei leider nicht möglich, beide Zähler zu verbinden, sodass nur anhand der Links die Quellen automatisch erstellt werden. Das ist aber auch nicht immer gewünscht, da es ggf. zweimal den selben Link gibt, der aber natürlich nur einmal als Quelle auftauchen soll.

Aus diesem Grund muss man die Quelllinks manuell am Ende nochmals einbinden, sodass jeder eindeutige (!) `\Link` sein `\Quellen`-Gegenstück hat:

```
\begin{Quellen}
\Quelle{http://www.ziel-adresse.de}
\end{Quellen}
```

Für die korrekte Trennung der URLs wird das Paket url benutzt.

Achtung: Diese Umsetzung ist nicht ganz sauber. Zum einen sollte man nicht vergessen, die Zähler zu Beginn jedes Artikels auf 1 zu setzen. Zum anderen passiert es dann aber, dass pdfTeX die doppelten Zähler moniert (*»pdfTeX warning (ext4): destination with the same identifier (namequellcounter.7) has been already used, duplicate ignored«*), die dann pro Artikel eindeutig sind, aber eben nicht über das gesamte Magazin gesehen.

Artikelumgebungen

Zu Beginn hatte ich einen Auszug des ursprünglichen Artikelanfangs gegeben. Dies wird inzwischen über die Umgebung

```
% Artikel mit Titel, Autor und PDF-Marke
\newenvironment{Artikel}[3]
{\begin{multicols}{3}[\Headline{#1}{#2}{#3}]}
{\end{multicols}}
```

gehandhabt, sodass der Artikelkopf per

```
\begin{Artikel}
{Magazinerstellung mit \LaTeX{}}
{Dominik Wagenf"uhr}
```

```
{2008_10_magazinerstellung}
TEXT
\end{Artikel}
```

eingebunden werden kann. \Headline ist dabei als

```
% Kopfzeile fuer Artikel
\newcommand{\Headline}[3]
{\pdfbookmark[2]{#1}{#3}\textbf{{\Large \hypertarget{#3}{#1}\label{#3}}\↩
  hspace*{4mm}von #2}\\}
```

definiert und erzeugt dabei PDF-Marke und Label und zeigt den Titel und den Autor an.

Autorenboxen

Eine recht neue Entwicklung sind die Autorenboxen, die man unter den meisten Artikeln in **freies**Magazin findet. Diese geben eine kurze Auskunft über den Autor und wie dieser zum Beispiel zu der Software steht, die er gerade vorgestellt hat. Die eigentliche Definition der Autorenbox ist sehr einfach:

```
% Autorenbox
\newenvironment{Autoreninfo}[3][3.3mm]
{\begin{fmBox}[#1]
{\textbf{Autoreninformation}}
{#2}{19.2mm}{0.75}{0.65}
\textbf{#3} }
{\end{fmBox}}
```

und kann dann über

```
\begin{Autoreninfo}{4}{Dominik Wagenf"uhr}
ist Redakteur bei \fm{} und freut sich, in der
TeXnische Kom"odie auftreten zu d"urfen.
\end{Autoreninfo}
```

eingebunden werden, wobei 4 die Anzahl der Zeilen für den Text angibt. Etwas komplizierter ist die Definition der \fmbox:

```
% neue Laengen
\newlength{\AbsGrafik}
\newlength{\AbsText}
\newlength{\BoHoehe}
```

```
\newenvironment{fmBox}[6][3.3mm]
{% Laengen setzen
\setlength{\BHoehe}
{-0.7\baselinestretch\baselineskip+1.1mm*\real{#3}+
\baselinestretch\baselineskip*\real{#3}*\real{0.8}}
\setlength{\AbsGrafik}{5.6mm+\BHoehe*\real{0.004}}
\setlength{\AbsText}{#4+\BHoehe*\real{0.993}}
% Hintergrundgrafiken einfuegen
\centering
\begin{minipage}{#5\linewidth}
  \includegraphics[width=\textwidth]{kopf.png}
  \vspace*{-\AbsGrafik}

  \includegraphics[width=\textwidth,height=\BHoehe]
  {mitte.png}
  \vspace*{-\AbsGrafik}

  \includegraphics[width=\textwidth]{fuss.png}
\end{minipage}
\vspace*{-\AbsText}

% Text schreiben
\begin{minipage}{#6\linewidth}
\hspace*{3.5mm}\textcolor{hpgrey}{#2}\\[#1]
\begin{footnotesize}}
{\\[0mm]
\end{footnotesize}
\end{minipage}
\vfill

}
```

Es werden dabei zuerst drei Abstände definiert, die sich nach der Anzahl der Zeilen richten, die man beim Aufruf von `\Autoreninfo` angibt. Die eigentliche Box besteht aus drei Bildern: einem Kopf, in dem die Überschrift steht, einer Mitte, die den Hintergrund für den Text bildet und zum Schluss einem Fuß, der die Box abschließt. Die definierten Abstände sind dabei wichtig, um die Grafiken und den enthaltenen Text korrekt auszurichten. Es wird also zuerst die Box als Grafik gezeichnet, dann mittels `\vspace` wieder im Text nach oben gesprungen und der eigentliche Text gedruckt.

Das ganze Konstrukt ist durch die `\vspace`-Benutzung und die empirische Berechnung der Abstände etwas wackelig, sodass es ein paar Nebeneffekte

gibt. Es ist z. B. nicht möglich, eine Box mit nur einer Zeile zu erstellen, weil die `\BoxHoehe` dann zu klein im Vergleich zu den eingebundenen Grafiken wird. An der Entwicklung dieser Box wird aber noch gearbeitet.

Das Endergebnis sieht in etwa so aus (normalerweise aber in Farbe):

Mobilversion

freiesMagazin wurde als PDF-Magazin konzipiert und wird auch dementsprechend geschrieben. Dennoch gab es einige Anfragen nach einer Version für Mobilgeräte, da das PDF darauf nur schlecht zu lesen ist. Aus diesem Grund hat man sich für eine Mobilversion in HTML entschieden. Voraussetzung allerdings war, dass dafür die Ausgabe nicht komplett neu geschrieben werden musste.

Auf der Suche nach einer Lösung stieß ich auf TTH, den »TeX to HTML translator« [5]. Das C-Programm ist zwar sehr mächtig, konnte aber dennoch nicht mit allen LaTeX-Befehlen etwas anfangen. Aber auch hier halfen die Befehlsdefinitionen in der Datei `befehle.tex` weiter, denn es wurde einfach eine zweite Version von `befehle.tex`, die `befehle-mobil.tex`, erstellt, die für die Erstellung der HTML-Ausgabe benutzt wird. Diese Mobilversion enthält keine komplizierten Umgebungen oder Tabellen, sondern versucht fast alles mit »LaTeX-Standardbefehlen« zu verwirklichen.

Als Beispiel: Aus der obigen `\fmbox` wurde diese Definition:

```
\newenvironment{fmBox}[6][3.3mm]
{\begin{Tabelle}{1}{|l|}{#2}}
{\\
\end{Tabelle}}
```

Wesentlich kompakter, nicht wahr? Diese Vereinfachung hilft aber dabei, die HTML-Version ohne jegliche Anpassung der eigentlichen Artikel zu erstellen.

Ausblick

Natürlich steht die Entwicklung der Befehlsdefinitionen nicht still. Ein offenes Problem ist z. B. die Erstellung des Inhaltsverzeichnisses. Dies geschieht aktuell zwar automatisch, aber leider ohne LaTeX-Hilfe. Für die Umsetzung wurde ein C++-Programm geschrieben, welches die Hauptdatei des Magazins durchgeht, aus den eingebundenen Artikeln die Titel extrahiert und das alles in eine neue LaTeX-Datei ausgibt. Eine vollautomatische LaTeX-Lösung im Stile eines `\tableofcontents` wäre traumhaft, konnte bisher aber nicht realisiert werden.

Zusammenfassung

Ich hoffe, dass der Artikel aufschlussreich war und gezeigt hat, wie man ein komplettes PDF-Magazin Monat für Monat mit LaTeX setzen kann, ohne auf Komfort oder ein gutes Layout verzichten zu müssen. Inzwischen kann man eigentlich sagen, dass die Setzer von **freies**Magazin im Endeffekt kaum noch LaTeX beherrschen müssen – schließlich gibt es für (fast) alles eine Befehlsdefinition.

Verweise

[1] ***freies**Magazin Extras – Logos und Quellen*; `http://www.freiesmagazin.de/extras/`.

[2] *Die **freies**Magazin-Website*; `http://www.freiesmagazin.de/`.

[3] *GNU Free Documentation License*; `http://www.gnu.org/copyleft/fdl.html`.

[4] *The TeX showcase*; `http://www.tug.org/texshowcase/`.

[5] *TTH: the TEX to HTML translator*; `http://hutchinson.belmont.ma.us/tth/`.

[6] *Das wrapfig-Paket*; `http://www.tex.ac.uk/tex-archive/help/Catalogue/entries/wrapfig.html`.

[7] *Using wrapfig to span multiple columns*; `http://www.tex.ac.uk/tex-archive/macros/latex/contrib/wrapfig/multiple-span.txt`.

Das Paket todonotes

Adelheid Grob

Schon vor einiger Zeit kam die Idee auf, in einer regelmäßigen Reihe kleinere TEX-Pakete vorzustellen, die entweder hilfreich sein können oder einfach Spaß machen. Dieser Artikel wird einen Einblick in die Möglichkeiten geben, die sich im Zusammenhang mit LATEX ergeben.

Einleitung und Anwendungsszenarien

Wer kennt das nicht? Man schreibt einen Artikel, eine Anleitung, eine Arbeit oder ein anderes längeres Dokument und an einigen Stellen fehlen noch Bilder, erläuternde Sätze oder gar ganze (Unter-)Kapitel.

In der Regel findet man an solchen Stellen oft Füllsätze wie »Baustelle«, »Bild XYZ« oder »Hier fehlt noch was«. Um solche Platzhalter übersichtlicher und effektiver gestalten zu können, findet man auf CTAN zwei Pakete: Zum einen das ältere `todo` von Frederico Garcia (2002), zum anderen das wesentlich leistungsstärkere `todonotes` von Henrik Skov Midtiby, welches ich hier vorstellen werde. Wie man den Beiträgen der CTAN-Mailingliste entnehmen kann, wurden diesem Paket in der letzten Zeit viele Erweiterungen hinzugefügt. Ich beziehe mich in diesem Artikel auf die Version vom 9. Januar 2009. Das Paket `todonotes` ist nicht nur hilfreich, um Platzhalter besser verwalten zu können, sondern kann auch beim Korrekturlesen gute Dienste leisten, sofern man die TEX-Quelle des zu korrigierenden Dokumentes vorliegen hat.

Einbinden, geladene Pakete, Abhängigkeiten, Optionen

Eingebunden wird `todonotes` über `\usepackage[Optionen]{todonotes}` in der Präambel. Es greift auf die folgenden Pakete zurück:

- `hyperref` um Verlinkungen zu entsprechenden Textstellen zu erzeugen,
- `xcolor` um das Verwenden von Farben zu ermöglichen,
- `tikz` um ein Beispielbild zu erzeugen,
- `xkeyval` um Zuordnungen in den Optionen wie z. B. `\bordercolor=orange` zu erlauben,
- `ifthen`,
- `calc`.

Diese Pakete werden automatisch geladen, sofern sie nicht bereits in der Präambel aufgeführt sind. Sofern in letzterem Fall bestimmte Optionen gewünscht sind, müssen die betreffenden Pakete *vor* `todonotes` geladen werden. Dem Paket `todonotes` können beim Laden folgende Optionen mitgegeben werden:

Option	*Beschreibung*	*Vorgabe*
`disable`	Hiermit können alle eingefügten Makros «ausgeschaltet« werden. Dies bietet sich an, wenn man sich seinen Text ohne ToDos anschauen möchte oder man alle ToDos abgearbeitet hat, diese aber nicht löschen kann oder will.	–
`SPRACHE`	Möchte man die Überschrift der ToDo-Liste oder die Bezeichnung eines Beispielbildes nicht auf englisch haben, so kann man mit dieser Option über die Parameter `danish`, `german`, `french`, `spanish` oder `catalan` eine andere Sprache wählen. Diese Liste wird ständig erweitert.	`english`
`colorinlistoftodos`	Mit dieser Option steuert man, ob man in der Liste der ToDos kleine farbige Quadrate vor dem eigentlichen Eintrag haben möchte. Dies bietet sich an, wenn man eingefügte ToDos z. B. nach Priorität farbig codiert hat.	–
`backgroundcolor`	Die eingefügten ToDos werden farbig hinterlegt. Mit der Option `\backgroundcolor` lässt sich diese Farbe näher bestimmen (Farbnamen von `xcolor` beachten!).	`orange`
`bordercolor`	Die Rahmenfarbe der Box, die um die einzelnen Kommentare gezogen wird.	`schwarz`
`linecolor`	Die Farbe der Linie, die zwischen der Box und der Textstelle, an der die Notiz eingefügt wurde, gezogen wird.	`orange`
`textwidth`	Breite der Box (für Notizen, die am Rand des Textes gesetzt werden; Boxen, die `\inline`, also im Textfluss gesetzt werden, haben stets Textbreite).	`\marginwidth`
`textsize`	Schriftgröße. Es sind die gängigen Größenbefehle, wie z. B. `\tiny`, möglich.	`\normalsize`
`prependcaption`	Diese Option ermöglicht, dass der `\caption`-Text, der in der Liste der ToDos auftaucht, in der Notiz selber erscheint.	–
`shadow`	Hiermit ist es möglich, die Box, die um eine Notiz gezogen wird, mit einem Schatten zu hinterlegen.	–

Option	*Beschreibung*	*Vorgabe*
`dvistyle`	Diese Option ist nötig, wenn man Dokumente mit `latex` übersetzt, da sonst der Text in den Boxen im `.dvi-File` nicht richtig gesetzt wird.	–

Befehlsreferenz

Das Paket `todonotes` stellt drei Befehle zur Verfügung: `\todo`, `\missingfigure` und `\listoftodos`.

`\todo`

Mit `\todo` fügt man eine neue Notiz ein. Ohne Optionen erscheint die Notiz am Seitenrand in einer Box mit der beim Laden eingestellten Hintergrundfarbe. Außerdem wird eine Verbindungslinie zwischen der Notiz und der Textstelle, an welcher das Kommando `\todo` aufgerufen wurde, hergestellt (siehe Beispiel). Mit zahlreichen Optionen kann dieses Verhalten geändert werden, beispielsweise ist es möglich, die Notiz im Text zu platzieren, die Linie wegzulassen oder farbliche Anpassungen zu machen. Die unten stehende Tabelle listet alle möglichen Optionen auf.

Option	*Beschreibung*
`color`	Hier können wie beim Einbinden des Pakets den Werten `backgroundcolor`, `linecolor` und `bordercolor` andere als die Standardfarbwerte zugewiesen werden, beispielsweise erzeugt `\todo[linecolor=red]{text}` eine Notiz, deren Verbindungslinie zum Text rot ist. Unterschiedliche Farbkodierungen werden, falls beim Laden des Pakets die Option `colorinlistoftodos` gesetzt wurde, in dieser durch kleine verschiedenfarbige Quadrate gekennzeichnet.
`line/noline`	Will man die Verbindungslinie ausschalten, kann man dies mit `\todo[noline]{text}` tun.
`inline/noinline`	`inline` erzeugt die Notiz innerhalb des Textes (siehe Beispiel), `noinline` ist die Standardeinstellung (setzt die Notiz in den Seitenrand).
`size`	Hiermit kann die Textgröße der Notiz angepasst werden, beispielsweise `\todo[size=\Large]{text}`.
`list/nolist`	Mit der Option `nolist` kann ein Eintrag einer bestimmten Notiz in der `\listoftodos` verhindert werden.

Option	*Beschreibung*
`caption`	Hiermit kann einer Notiz eine Unterschrift (caption) mitgegeben werden. Sie erscheint dann in der `\listoftodos`, beispielsweise `\todo[caption={testnote}]{text}`. Diese Option bietet sich für sehr lange ToDo-Notes an.
`prepend/noprepend`	Diese Option hängt mit der vorherigen zusammen. Mit `prepend` wird die Unterschrift (caption) auch in der Todo-Note selber angezeigt (siehe Beispiel).

`\missingfigure`

Mit dem Befehl `\missingfigure` kann ein Beispielbild eingefügt werden (siehe Beispiele). Dieses kann mit einem erläuternden Text versehen werden. Weiterhin kann man die Weite des eingefügten Platzhalters mit der Option `figwidth` angeben. Wird diese Option freigelassen, erstreckt sich das Bild über die gesamte Textbreite. Gibt man beim Einbinden des Pakets die Option `german` an, so erscheint der Text im Bild auf Deutsch.

`\listoftodos`

Mit dem Befehl `\listoftodos` lässt sich eine Liste aller Todos im Text aufführen.

Ausblick auf künftige Entwicklung

Wie bereits in der Einleitung erwähnt, tut sich in der Entwicklung in diesem Paket momentan recht viel. Angedacht sind zusätzliche Optionen beim Befehl `\todo`, wie z. B. `owner`, um anzugeben, wer die Notiz erstellt hat, `due=2009-02-02` für die Angabe eines Zeitpunkts, bis wann die `todonote` erledigt sein muss oder die Option `done`, um abgearbeitete `todonotes` zu kennzeichnen.

Bugs & Fleas

Verwendet man statt pdfLaTeX aus bestimmten Gründen LaTeX, werden in der `\todolist` lange Zeilen nicht umbrochen, sondern laufen über den Rand hinaus, was man bei der Angabe der `\caption` berücksichtigen sollte.

Leider arbeitet das Paket mit der Dokumentenklasse `dtk` nicht gut zusammen, da der Rand zu schmal ist.

Beispiele

Dies ist ein Minimalbeispiel, um zu zeigen, wie `todonotes` funktioniert; geladen wurde das Paket mit den Optionen `prependcaption, german`.

Das ist ein Test, um zu zeigen, wie eine einfache ToDo-Note aussieht. Der eingefügte Befehl lautet: `\todo[]{das ist eine einfache ToDo-Note}`

das ist eine einfache ToDo-Note

Das ist ein Test, um zu zeigen, wie eine ToDo-Note aussieht, die von einer andersfarbigen Box umschlossen wird. Der eingefügte Befehl lautet:
`\todo[backgroundcolor=green!40]{ToDo grüner Box und grüner Linie}`

ToDo mit grüner Box und grüner Linie

Das ist ein Test, um zu zeigen, wie eine ToDo-Note aussieht, die keine Linie zur entsprechenden Textstelle aufweist. Der eingefügte Befehl lautet:
`\todo[noline]{ToDo ohne Linie zur Textstelle}`

ToDo ohne Linie zur Textstelle

Das ist ein Test, um zu zeigen, wie

ToDo-Note, die im Text statt am Rand erscheint

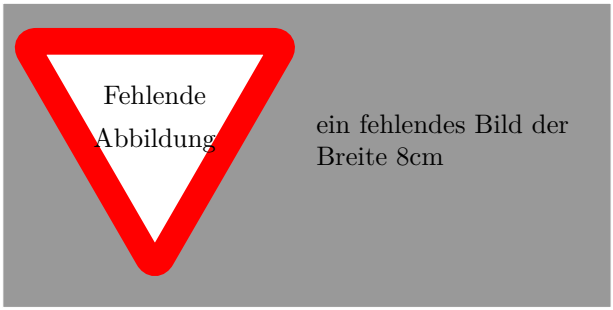
eine ToDo-Note aussieht, die im Fließtext erscheint. Der eingefügte Befehl lautet:
`\todo[inline]{ToDo-Note, die im Text statt am Rand erscheint}`

Das ist ein Test, um zu zeigen, wie eine ToDo-Note aussieht, die eine Caption enthält, die in der Note genannt wird. Der eingefügte Befehl lautet:
`\todo[prepend,caption={test}]{ToDo mit Caption und prepend}`

test: ToDo mit Caption und prepend

Ein fehlendes Bild, eingefügt mit `\missingfigure{Ein fehlendes Bild}`

Ein fehlendes Bild, eingefügt mit
`\missingfigure[figwidth=8cm]{ein fehlendes Bild der Breite 8cm}`



Spielereien mit TEX-Logos in HTML

Gerd Neugebauer

Auch wenn TEX und seine Freunde fast perfekte Ergebnisse im Print-Bereich erzeugen können, so ist der Durchbruch im Web bisher nicht gelungen. Also gibt es im Web immer noch viele Seiten, die mit HTML gesetzt sind. Wenn man nun etwas über TEX im Web schreibt, dann ergibt sich automatisch die Frage, wie man das Logo am besten setzt. In diesem Artikel werden hierzu einige Antworten gegeben.

Einleitung

Die erste Antwort, wie man das TEX-Logo setzt, stammt natürlich von Donald Knuth. Im TEXbook [1] wird die folgenden Definition gegeben:

```
\def\TeX{T\kern-.2em\lower.5ex\hbox{E}\kern-.06em X}
```

Daneben wird angegeben, dass als Ersatzdarstellung `TeX` genommen werden sollte, wenn man das Logo nicht als TEX setzen kann. Das ist zumindest für das Web eine erste Variante, wenn auch nicht besonders ansprechend. Mit dem Logo ist schließlich ein gewisser Wiedererkennungswert verbunden, der in der Textvariante etwas verloren geht.

Eine weitere Variante für das Web besteht darin, eine kleine Grafik einzubinden, um das TEX-Logo in seiner vollen Schönheit zu zeigen. Das hat aber einige Nachteile. So ist der Text nicht suchbar – und wer will schon von Google nicht gefunden werden? Weiterhin passt sich die Grafik nicht so ohne weiteres an die vorhandenen Schriftgrößen und Schriftarten an. Wenn man die Schriftart also nicht unverrückbar festnagelt, dann besteht die Gefahr, dass das Logo als Fremdkörper erscheint. Diese »Lösung« ist also nicht sehr attraktiv.

Einfache HTML-Mittel

Als erste Annäherung an eine Lösung sehen wir uns einige einfache Mittel an, die HTML uns zur Verfügung stellt. Ein solches Mittel ist das HTML-Element `sub`. Dieses setzt sein Argument als Subskript – also etwas tiefer und etwas kleiner. In HTML sieht das dann folgendermaßen aus:

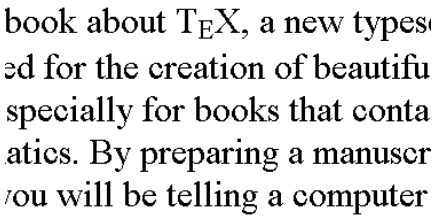


Abbildung 1: Schlechte Approximation mit `<sub>`

```
T<sub>E</sub>X
```

Das Ergebnis ist in der Abbildung 1 dargestellt.[1] Es ist offensichtlich, dass diese Lösung unserem Anspruch nicht gerecht werden kann. Das E ist sichtbar zu klein und steht deutlich vom voran stehenden T ab.

Exkurs: horizontale und vertikale Abstände

Wenn wir in TEX Längen angeben, dann können hierzu eine Vielzahl von Längeneinheiten benutzt werden. Da kann dann beispielsweise »42mm« stehen. In Makros hat das den Nachteil, dass immer diese Länge benutzt wird, selbst wenn sich die Schriftgröße ändert. Deshalb ist es manchmal sinnvoll, Längenangaben zu verwenden, die in Abhängigkeit von der aktuellen Schriftgröße definiert sind.

Hierzu gibt es zwei Längenangaben em und ex. Die Länge 1em entspricht in etwa der Breite des Buchstabens m der aktuellen Schrift. Diese Breite hängt vom Schrift-Design ab und spiegelt eine horizontale Breite wieder. Breite Schriften haben einen größeren Wert als schmalere Schriften. Bei einer Verwendung einer horizontalen Länge in der Einheit em passt sich die Größe an die Breite der aktuellen Schrift an.

Die Längenangabe ex ist das vertikale Analogon. Es entspricht der Höhe des Buchstabens x in der aktuellen Schrift. Höhenangaben in der Einheit ex passen sich an die Höhe des aktuellen Fonts an.

In Cascading Style Sheets (CSS) kann man ebenfalls die Einheiten ex und em verwenden. Damit gibt es Möglichkeit, Definitionen zu erstellen, die sich an die aktuelle Schrift anpassen.

[1] Als Beispieltext dient der erste Abschnitt aus [1].

Schiebereien in CSS

HTML bietet die Möglichkeit, über CSS Eigenschaften von Elementen zu bestimmen. Ähnlich wie in TEX können diese Eigenschaften an Boxen angebracht werden. Das Pendant zu `\mbox` ist dabei das Element `<span>`. Über das Attribut `class` kann eine Klasse vergeben werden, die dann die entsprechenden Eigenschaften definiert. Da wir nur das E etwas verschieben wollen, schließen wir dieses in ein `<span>` ein. Das sieht dann folgendermaßen aus:

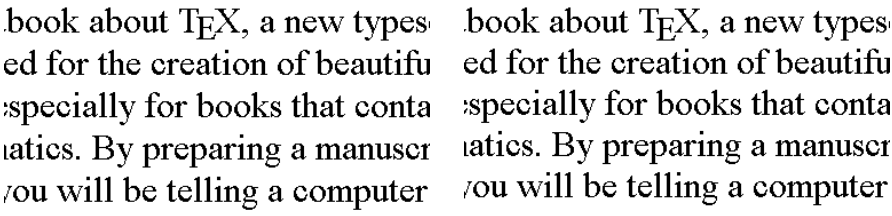
```
T<span class="e">e</span>X
```

Nun können wir noch die Eigenschaften der Klasse e definieren. Dies geschieht in einer externen CSS-Datei, die eingebunden wird, oder einfach durch Einbetten in die HTML-Datei selbst. Dies geschieht im Kopf – also zwischen `<head>` und `</head>`.

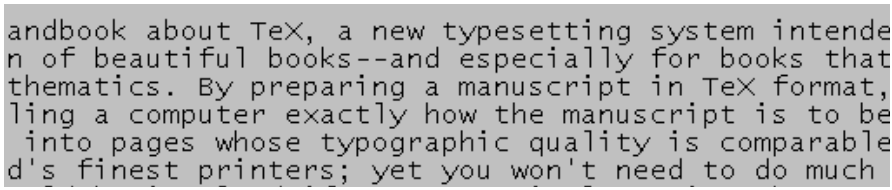
Welche Eigenschaften können wir benutzen?

- Als erstes müssen wir das E etwas nach unter verschieben. Hierzu gibt es die Eigenschaft `vertical-align`, die eine vertikale Verschiebung bewirkt.
- Als nächstes emulieren wir die Unterschneidung (kerning) auf der linken Seite. Hierzu könne wir die Eigenschaft `margin-left` verwenden. Ein negativer Wert verschiebt nach links.
- Weiterhin brauchen wir die Unterschneidung auf der rechten Seite. Hier kommt die Eigneschaft `margin-right` zum Einsatz.
- Einige Experimente haben ergeben, dass das tiefer gestellte E in den niedrigen Auflösungen der Web-Browser recht wuchtig wirkt. Deshalb habe ich es dezent verkleinert. Das geht mit der Eigenschaft `font-size`. Ein Abschlag von 5% macht es etwas gefälliger, ohne unangenehm ins Auge zu fallen.
- Das nach unten versetzte E führt dazu, dass der Zeilenabstand beeinflusst wird. Zeilen mit dem TEX-Logo haben einen etwas größeren Durchschuss als solche ohne Logo. Mit der Eigenschaft `line-height` kann das beeinflusst wird. Mit einem Wert von 0 verschwindet der negative Effekt.
- Es ist sicher schon aufgefallen, dass in der Definition das e und nicht das E verwendet wurde. Das kann durch die Eigenschaft `text-transform` und den Wert `uppercase` korrigiert werden. Dadurch bleibt das bekannte Aussehen des Logos beim Entfernen von HTML-Code erhalten. Das Gleiche ist auch in dem Text-Browser Lynx zu beobachten (siehe Abbildung 2c).

Die Definition der Eigenschaften sieht dann folgendermaßen aus:



(a) Firefox 3 (b) Internet Explorer 6



(c) Lynx

Abbildung 2: Darstellung im Browser

```
<style type="text/css">
.e {
  text-transform:uppercase;
  font-size:95%;
  vertical-align: -0.45ex;
  margin-left: -0.1em;
  margin-right: -0.06em;
  line-height:0;
}
</style>
```

Das Ergebnis in verschiedenen Browsern ist in der Abbildung 2 zu sehen.

Weitere Logos

Nachdem wir die Grundlagen haben, können wir einige weitere Logos auf ähnliche Weise umsetzen. Recht einfach it ε-TEX. Hier müssen wir nur das Epsilon vor das TEX packen, das wir schon haben. Das Unterschneiden des T geht wieder mit der selben Methode, die wir schon diskutiert haben.

```
<style type="text/css">
.e {
  text-transform:uppercase;
  font-size:95%;
  vertical-align: -0.45ex;
  margin-left: -0.1em;
  margin-right: -0.06em;
  line-height:0;
}
.t {
  margin-left: -0.1em;
}
</style>
```

```
&epsilon;-<span class="t">T</span></span><span class="e">e</span>X
```

Die eben gezeigten Definitionen führen zu dem Ergebnis aus der Abbildung 3.

ε-TEX

Abbildung 3: Das ε-TEX-Logo in Firefox 3

Als nächstes setzen wir das BibTEX-Logo. Hier kommt hinzu, dass die ersten drei Buchstaben in Kapitälchen gesetzt sind. Dafür gibt es die Angabe der CSS-Eigenschaft »font-variant« mit dem Wert `small-caps`. Falls Kapitälchen nicht zur Verfügung stehen werden diese von den aktuellen Browsern emuliert.

```
<style>
.e {
  text-transform:uppercase;
  font-size:95%;
  vertical-align: -0.45ex;
  margin-left: -0.1em;
  margin-right: -0.06em;
  line-height:0;
}
.t {
  margin-left: -0.1em;
}
.b {
  font-variant: small-caps;
```

```
}
</style>
```

```
<span class="b">Bib</small><span
  class="t">T</span><span class="e">e</span>X
```

Die eben gezeigten Definitionen führen zu dem Ergebnis aus der Abbildung 4.

BIBTEX

Abbildung 4: Das BIBTEX-Logo in Firefox 3

Schließlich wollen wir uns noch an das LATEX-Logo wagen. Das Problem dabei ist das A. Dieses ist kleiner und über das L geschoben. Die hierfür benötigten Eigenschaften haben wir aber schon gesehen und müssen sie jetzt nur noch einsetzen: Transformation von Kleinbuchstabe zu Großbuchstabe, Verschieben nach Links und Verschieben nach oben. Das sieht dann folgendermaßen aus:

```
<style>
.e {
  text-transform:uppercase;
  font-size:95%;
  vertical-align: -0.45ex;
  margin-left: -0.1em;
  margin-right: -0.06em;
  line-height:0;
}
.a {
  text-transform:uppercase;
  font-size:75%;
  vertical-align: 0.45ex;
  margin-left: -0.36em;
  margin-right: -0.15em;
}
</style>
```

```
L<span class="a">a</span>T<span class="e">e</span>X
```

Die eben gezeigten Definitionen führen zu dem Ergebnis aus der Abbildung 5.

Abschließend zeigen wir in der Abbildung 6 noch einige Logos in einer serifenlosen Schrift. Die Definitionen sind die gleichen, die wir weiter oben

Abbildung 5: Das LaTeX-Logo in Firefox 3

bereits gesehen haben. Damit haben sich die Definitionen für diese Schrift auch bewährt.

Abbildung 6: Logos in Grotesktschrift

Schluß

Wir haben gesehen, dass man mit einigen kleinen Kniffen beim Setzen von Logos auch im Web mit HTML mit einigermaßen befriedigenden Ergebnissen aufwarten kann. Es besteht also keine Notwendigkeit, sich zu verstecken. Wir können die Logos auch im Web zeigen und damit zu einer Wiedererkennung auch in diesem Medium kommen.

Das gezeigte Verfahren hat natürlich auch seine Grenzen. Das ist dann der Fall, wenn für die Logos spezielle Schriften eingesetzt werden. Dies ist beispielsweise bei METAFONT der Fall. Hier müssen wir bisher noch aufgeben.

Literatur

[1] Donald E. Knuth: *The TEXbook*; Bd. A von *Computers and Typesetting*; Addison-Wesley Publishing Company; Reading, Mass.; 15. Aufl.; 1989.

Von fremden Bühnen

Neue Pakete auf CTAN

Jürgen Fenn

Der Beitrag stellt neue Pakete auf CTAN seit der letzten Ausgabe bis zum Redaktionsschluss vor. Die Liste folgt der umgekehrten chronologischen Reihenfolge. Bloße Updates werden nicht aufgeführt. Sie können auf der moderierten *tex-announce*-Mailingliste verfolgt werden, die auch online unter `http://blog.gmane.org/gmane.comp.tex.ctan.announce` verfügbar ist.

`acroreloadpdf` von *Alexander Grahn* erweitert das »File«-Menü von Adobe Reader unter Linux und auf anderen Unix-Plattformen um einen Eintrag »Reload«, mit dem man das aktuell geöffnete Dokument neu laden kann. Dabei bleiben alle Einstellungen erhalten.
`CTAN:support/acroreloadpdf`

`mathesatz-examples` von *Herbert Voß* enthält die Beispiele aus seinem Buch »Mathematiksatz mit LaTeX«.
`CTAN:info/examples/Math`

`schemabloc` von *Robert Papanicola* ist ein pgf/TikZ-Paket zum Zeichnen von Blockdiagrammen.
`CTAN:graphics/pgf/contrib/schemabloc`

`metalogo` von *Andrew Gilbert Moschou* enthält Parameter, mit denen man die diversen TeX-Logos an verschiedene Fonts anpassen kann, speziell für XeLaTeX-Anwender.
`CTAN:macros/latex/contrib/metalogo`

`pst-sigsys` von *Farshid Delgosha* ist eine Sammlung von PSTricks-Makros für Darstellungen zur Signalverarbeitung.
`CTAN:graphics/pstricks/contrib/pst-sigsys`

`sageep` von *Boris Veytsman* ist ein LaTeX-Stil für die *Environmental and Engineering Geophysical Society's Annual Meeting Papers*.
`CTAN:macros/latex/contrib/sageep`

`xecolour` von *Vafa Khalighi* definiert ungefähr 140 Farben, die mit XeTeX im bidirektionalen Textsatz verwendet werden können.
`CTAN:macros/xetex/latex/xecolour`

`ctanify` von *Scott Pakin* ist ein Perl-Skript, das aus `ins`-, `dtx`-, `pdf`- und README-Dateien ein `tar.gz`-Archiv nach den Vorgaben unter `http://www.ctan.org/upload.html` erstellt, das man auf CTAN hochladen kann. Das Archiv enthält auch ein `tds.zip`-Archiv zur leichten händischen Installation. Das Skript wurde unter Linux getestet und läuft auch unter Mac OS X.
`CTAN:support/ctanify`

`zwpagelayout` von *Zdenek Wagner* dient zum Festlegen des Seitenlayouts und zum Erzeugen von Schneidemarken (*cropmarks*) mithilfe von TEX zusammen mit dvips oder (x)dvipdfm(x) oder mit pdftex allein. Außerdem können damit Seiten horizontal und vertikal gespiegelt werden.
`CTAN:macros/latex/contrib/zwpagelayout`

`tdclock` und `analogclock` von *Luis Randez* erzeugt mit pdflatex »live« tickende digitale und analoge Uhren, die das aktuelle Datum und die Systemzeit beim Öffnen des Dokuments im Adobe Reader anzeigen. Die Darstellung der Analoguhr funktioniert leider nur unter MS Windows.
`CTAN:macros/latex/contrib/tdclock`
`CTAN:macros/latex/contrib/analogclock`

`induni-om` von *John Smith* enthält einen virtuellen Font für Omega im Unicode-Format mit vielen Sonderzeichen, insbesondere für Sanskrit.
`CTAN:fonts/induni-om`

`emptypage` von *Karl Wette* dient zum Unterdrücken der Seitenzahlen und der Kopfzeilen bei leeren Seiten eines Dokuments.
`CTAN:macros/latex/contrib/emptypage`

`biblatex-jura` von *Ben E. Hard* ist ein `biblatex`-Stil für Zitate in rechtswissenschaftlichen Arbeiten nach den Vorgaben des Nomos-Verlags.
`CTAN:macros/latex/exptl/biblatex-contrib/biblatex-jura`

`dozenal` von *Donald P. Goodman III* erlaubt es, Zahlen (auch die Standardzähler) im Duodezimalsystem (also zur Basis 12) zu setzen. Ein Makro von *David Kastrup* erlaubt die Konvertierung vom Dezimal- ins Duodezimalsystem. Fonts für duodezimale Werte, die gut zur *Computer Modern* passen, sind beigefügt.
`CTAN:fonts/dozenal`

`pdfcomment` von *Josef Kleber* dient zum Einfügen von Anmerkungen in PDF-Dokumente, die mit Adobe Reader betrachtet werden können.
`CTAN:macros/latex/contrib/pdfcomment`

`pigpen` von *Oliver Corff* dient zum Setzen des Freimaurer-Quadrats, einer historischen monoalphabetischen Substitutionschiffre.
`CTAN:fonts/pigpen`

`pdfx` von *CV Radhakrishnan* und *Hàn Thế Thành* bietet PDF/X-1a- und PDF/A-1b-Unterstützung für pdfTEX.
`CTAN:macros/latex/contrib/pdfx`

`biblatex-apa` von *Philip Kime* ist ein `biblatex`-Stil, der die aktuellen Richtlinien *American Psychological Association* umsetzt.
`CTAN:macros/latex/exptl/biblatex-contrib/biblatex-apa`

`pst-bspline` von *Michael Sharpe* dient zum Zeichnen von B-Spline-Kurven mithilfe von PSTricks.
`CTAN:graphics/pstricks/contrib/pst-bspline`

`fahyph` von *Vafa Khalighi* sind persische Trennmuster für XETEX.
`CTAN:language/hyphenation/fahyph`

`easylist` von *Paul Isambert* dient zum Erzeugen von Listen, wie man sie aus *Wittgensteins »Tractatus logico-philosophicus«* kennt.
`CTAN:macros/latex/contrib/easylist`

`gentium` von *Thomas A. Schmitz* und *Mojca Miklavec* enthält die TrueType-Schriftart Gentium nebst

LaTeX- und ConTeXt-Unterstützung.
CTAN:fonts/gentium

`tex-ewd` von *Wolfgang Helbig* dient zum Setzen von mathematischen Beweisen und von Programmen im Stile der *guarded command language* von *Dijkstras* .
CTAN:macros/generic/tex-ewd

`psbao` von *Nino Vessella* dient zum Zeichnen von Diagrammen für das Spiel Bao mithilfe von `pstricks`.
CTAN:graphics/pstricks/contrib/psbao

`ionumbers` von *Christian Schneider* kann Zahlen im Mathemathikmodus von angelsächsischer (»1,234.56«) zu deutscher Notation (»1.234,56«) umsetzen (und umgekehrt).
CTAN:macros/latex/contrib/ionumbers

`lithuanian` von *Sigitas Tolušis* enthält die Trennmuster und diverse Ergänzungen für Texte in Litauisch.
CTAN:language/lithuanian

`vaucanson-g` von *Sylvain Lombardy* und *Jacques Sakarovitch* dient zum Zeichnen von Automaten mit PSTricks.
CTAN:graphics/pstricks/contrib/vaucanson-g

`arsclassica` von *Lorenzo Pantieri* stellt einige Änderungen zu der Klasse `classicthesis` von *André Miede* zur Verfügung.
CTAN:macros/latex/contrib/arsclassica

`shuffle` von *Antoine Lejay* und *J. Gilbey* stellt das methematische Symbol »Shuffle-Produkt« in METAFONT bereit.
CTAN:fonts/shuffle

`cfr-lm` von *Clea F. Rees* ist eine verbesserte LaTeX-Unterstützung für die Latin-Modern-Schriften (*Oldstyle*-Ziffern, Kapitälchen, *Upright*-Kursive, diverse alternative Schnitte, proportionale Schreibmaschinenschrift und den »Quotation«-Font).
CTAN:macros/latex/exptl/cfr-lm

`syllogism` von *Nicolas Vaughan* dient zum Setzen von Syllogismen mit LaTeX.
CTAN:macros/latex/contrib/syllogism

`historische-zeitschrift` von *Dominik Wassenhoven* ist ein `biblatex`-Stil für die gleichnamige Zeitschrift.
CTAN:macros/latex/exptl/biblatex-contrib/historische-zeitschrift

`biblatex-nature` von *Joseph Wright* ist ein `biblatex`-Stil für *Nature*.
CTAN:macros/latex/exptl/biblatex-contrib/biblatex-nature

`context-account` von *Wolfgang Schuster* dient zum Setzen von T-Konten mit ConTeXt, wie sie z. B. zur Darstellung einer Bilanz benötigt werden.
CTAN:macros/context/contrib/context-account

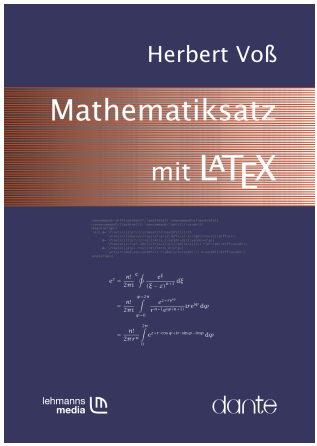
`context-inifile` von *Peter Münster* dient zum formatierten Ausdrucken von Ini-File-Quelltext (*pretty printing*) als Tabelle mithilfe von ConTeXt.
CTAN:macros/context/contrib/context-inifile

`context-letter` von *Wolfgang Schuster* dient zum Setzen von Briefen mithilfe von ConTeXt.
CTAN:macros/context/contrib/context-letter

Bücher und Rezensionen

Edition dante – Neuerscheinung

Herbert Voß:
Mathematiksatz mit LaTeX;
DANTE e. V. und Lehmanns Media
304 Seiten (ISBN 978-3-86541-319-2;
19,95 € (Ladenpreis) bzw. 15,– € für Mitglieder
von DANTE e.V., jeweils versandkostenfrei)



Bestellung

Bitte schicken Sie eine E-Mail an `office@dante.de` mit Angabe von *Name, Anschrift, Mitgliedsnummer* und *Anzahl der Exemplare*, und überweisen Sie den Betrag auf das Konto von DANTE e.V. oder bezahlen Sie den Betrag per PayPal. Die Kontonummer finden Sie am Ende dieses Heftes und Informationen zu PayPal auf `http://www.dante.de/dante/zahlung/`.

Bitte beachten Sie für Bestellungen bei DANTE e.V. folgende Informationen zum Widerrufsrecht: Verbraucher können bei Bestellungen per E-Mail, Internet, Brief oder Telefon den Kaufvertrag innerhalb einer Frist von 14 Tagen ab Erhalt der Ware per Brief, Fax oder E-Mail oder durch Rücksendung der Ware widerrufen (siehe Kontaktadresse). Zur Wahrung der Frist genügt die rechtzeitige Absendung des Widerrufs oder der Ware. Bei einem Bestellwert bis 40,– € hat der Besteller die Rücksendekosten zu tragen. Bei Verschlechterung der Ware, die über die übliche Prüfung der Ware hinausgeht, hat der Besteller gegebenenfalls Wertersatz zu leisten.

Die TEXnische Komödie

dante
Deutschsprachige
Anwendervereinigung TEX e.V.

21. Jahrgang Heft 1/2009 Februar 2009

1/2009

Die TEXnische Komödie

dante
Deutschsprachige
Anwendervereinigung TEX e.V.

21. Jahrgang Heft 1/2009 Februar 2009

1/2009

Spielplan

Termine

2009

25. 2. – 27. 2. **DANTE 2009**
und 40. Mitgliederversammlung von DANTE e.V.
Technischen Universität Wien
`http://www.dante.de/dante/events/dante2009/`

29. 4. – 3. 5. **17. BachoTEX-Konferenz BachoTEX 2009**
Bachotek, nahe Brodnica, Polen
`http://www.gust.org.pl/BachoTeX/2009`

28. 7. – 31. 7. **TUG 2009**
Notre Dame (Indiana), USA
`http://tug.org/tug2009/`

24. 8. – 28. 8. **EuroTEX 2009 und ConTEXt meeting**
Den Haag, Niederlande
`http://tug.org/tug2009/`

Die Abbildung auf der folgenden Seite ist das offizielle Poster für die TEX-Tagung DANTE 2009 in Wien.

Umgebungen

```
$...$                    \(...\)                    \[...\]
\begin{abstract} ...\end{abstract}
\begin{array} [Position] {Spaltendefinition}
  ...&...&...&...\\
     ...
\end{array}
\begin{center} ...\end{center}
\begin{description}
  \item[Label] Text
     ...
\end{description}
\begin{displaymath} ...\end{displaymath}
\begin{document} ...\end{document}
\begin{enumerate}
  \item [#] Text
     ...
\end{enumerate}
\begin{eqnarray * }
  ...&...&...\\
     ...
\end{eqnarray * }
\begin{equation} ...\end{equation}
\begin{figure * } [Pos] ...\end{figure * }
\begin{filecontents * }{Dateiname}
    ...
\end{filecontents * }
\begin{flushleft} ...\end{flushleft}
\begin{flushright} ...\end{flushright}
\begin{itemize}
  \item [Symbol] Text
     ...
\end{itemize}
\begin{list}{Label}{Parameter}
  \item [...] Text
     ...
\end{list}
\begin{lrbox}{\<Boxname>} ...\end{lrbox}
\begin{math} ...\end{math}
\begin{minipage} [vPos] [Höhe] [iPos] {Breite}
  ...
\end{minipage}
\begin{picture}(x,y) (dx,dy) ...\end{picture}
\begin{quotation} ...\end{quotation}
\begin{quote} ...\end{quote}
\begin{sloppypar} ...\end{sloppypar}
\begin{tabbing} ...\end{tabbing}
\begin{table * } [Pos] ...\end{table * }
\begin{tabular} [Position] {Spaltendef.}
  ...&...&...&...\\
     ...
\end{tabular}
\begin{tabular*}{Breite} [Pos.] {Spaltendef.}
  ...&...&...&...\\
     ...
\end{tabular*}
\begin{thebibliography}{Labelbreite}
  \bibitem [Label] {Schlüssel}Text
  ...
\end{thebibliography}
\begin{theindex}
  \item Text
    \subitem Text
      \subsubitem Text
    ...
\end{theindex}
\begin{titlepage} ...\end{titlepage}
\begin{trivlist} ...\end{trivlist}
\begin{verbatim * } ...\end{verbatim * }
\begin{verse} ...\end{verse}
```

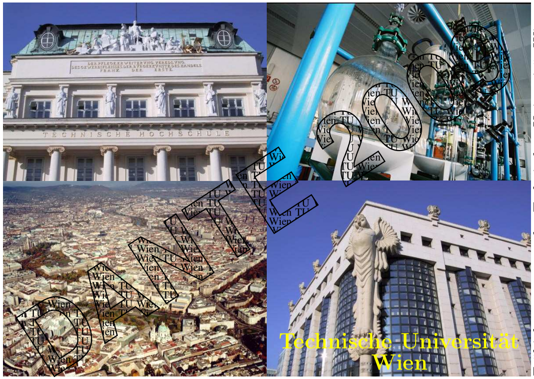
Befehle und aktive Zeichen

```
{           \:     \'     \#     \{
}           \!     \`     \$     \AA
~           \-     \|     \%     \aa
\<space>    \=     \(     \&
\@          \>     \)     \~
\\ * [Länge]       \[     \_
\,          \<     \]     \^
\;          \+     \/     \}
```

```
\addcontentsline{Dateityp}{Ebene}{Eintrag}
\addtocontents{Dateityp}{Eintrag}
\addtocounter{Zähler}{Wert}
\addtolength\<Längenmakro>{Länge}
\addvspace{Länge}
\Alph{Zähler}
\alph{Zähler}
\and
\appendix
\arabic{Zähler}
\AtBeginDocument{Argument}
\AtBeginDvi{Argument}
\AtEndDocument{Argument}
\AtEndOfPackage{Argument}
\AtEndOfClass{Argument}
\author{Name}
\backslash
\baselinestretch
\bfdefault
\bfseries
\bgroup
\bibitem [Label] {Key}
\bibliography{Dateiname}
\bibliographystyle{Stil}
\bigskip
\boldmath
\caption [Verzeichnis] {Unterschrift}
\cdots
\centering
\chapter * {Titel}
\chaptermark{Argument}
\circle * {Durchmesser}
\cite [Text] {Key(s)}
\cleardoublepage
\clearpage
\cline{i-j}
\dag
\ddag
\dashbox{Dash-Länge}(w,h) [Pos] {Inhalt}
\date{Text}
\ddots
\DeclareErrorFont
  {encoding}{family}{series}{shape}{size}
\DeclareFixedFont{\<Name>}
  {encoding}{family}{series}{shape}{size}
\DeclareFontEncoding{Name}{Text}{Mathe}
\DeclareFontEncodingDefaults{Text}{Mathe}
\DeclareFontFamily{enc.}{family}{Optionen}
\DeclareFontShape{encoding}{family}{series}
  {shape}{Liste}{Optionen}
\DeclareFontSubstitution
  {encoding}{family}{series}{shape}
\DeclareMathAccent{\<Name>}{\<Typ>}
  {Font}{Nummer}
\DeclareMathAlphabet{\<Name>}
  {encoding}{family}{series}{shape}
\DeclareMathDelimiter{\<Name>}
  {Typ}{Normal}{Nummer}{Groß}{Nummer}
\DeclareMathRadical{\<Name>}
  {Normal}{Nummer}{Groß}{Nummer}
\DeclareMathSizes{textsize}{displaysize}
  {scriptsize}{scriptscriptsize}
\DeclareMathSymbol{Name}{Typ}{Font}{Nummer}
\DeclareMathVersion{Version}
\DeclareOption*{Name}{Code}
\DeclareRobustCommand\<Makroname>{Definition}
\DeclareSymbolFont{Name}
  {encoding}{family}{series}{shape}
\DeclareSymbolFontAlphabet{\<Makroname>}{Font}
\displaystyle
\documentclass [Optionen] {Klassenname}
\dotfill
\egroup
\emph
\encodingdefault
\endgraf
\endline
\enlargethispage*{Länge}
\enspace
\enskip
\ensuremath{Ausdruck}
\ExecuteOptions{Optionen}
\familydefault
\fbox{Text}
\flushbottom
\fmtname
\fmtversion
\fnsymbol{Zähler}
\fontencoding{Encoding}
\fontfamily{Familie}
\fontseries{Serie}
\fontshape{Shape}
\fontsize{Größe}{Zeilenabstand}
\footnote [Nummer] {Text}
\footnotemark [Nummer]
\footnoterule
\footnotesize
\footnotetext [Nummer] {Text}
\frac{Zähler}{Nenner}
\frame{Text}
\framebox [Breite] [Position] {Text}
\framebox(Breite,Höhe) [Position] {Text}
\frenchspacing
\fussy
\hangindent{Länge}
\hfill
\hline
\hrulefill
\hspace*{Länge}
\huge
\Huge
\hyphenation{Wortliste}
\IfFileExists{Dateiname}{Ja-Aktion}{Nein-Aktion}
\include{Datei}
\includeonly{Datei1,Datei2,...}
\indent
\input{Datei}
\InputIfFileExists{Dateiname}
  {Ja-Aktion}{Nein-Aktion}
\itdefault
\item [Label]
\itshape
\kill
\label{Key}
\large
\Large
\LARGE
\LaTeX
\LaTeXe
\lbrack
\ldots
```

dante

Deutschsprachige Anwendervereinigung
TeX e. V., Heidelberg



Frühjahrstagung an der Technischen Universität Wien
25. bis 27. Februar 2009 (http://www.dante.de/dante2009)

```
\left{Klammertyp}
\leftmark
\let\<MakroA>=\<MakroB>
\limits
\line(dx,dy){Länge}
\linebreak [Wert]
\linethickness{Dicke}
\listfiles
\llap{Text}
\LoadClass [Optionen] {Dokum.-klasse} [Datum]
\LoadClassWithOptions{Dokum.-klasse} [Datum]
\loop{Schleifenargumente}
\lq
\makeatletter
\makeatother
\makebox [Breite] [Position] {Text}
\makebox(Breite,Höhe) [Position] {Text}
\makeindex
\MakeLowercase{Text}
\MakeRobust\<Makro>
\maketitle
\MakeUppercase{Text}
\marginpar [Links] {Rechts}
\markboth{Linker Kopf}{Rechter Kopf}
\markright{Rechter Kopf}
\mathbf{Math}
\mathcal{Math}
\mathit{Math}
\mathnormal{Math}
\mathop{Operator}
\mathrm{Math}
\mathsf{Math}
\mathtt{Math}
\mathversion{Version}
\mbox{Text}
\mddefault
\mdseries
\medskip
\multicolumn{AnzSpalten}{Pos}{Text}
\multiput(x,y)(dx,dy){N}{Objekt}
\narrower
\NeedsTeXFormat{Format} [Datum]
\negthinspace
\newcommand * \<Name> [N] [Standard] {Def.}
\newcounter{Name} [Rücksetzzähler]
\newenvironment{Name} [N] [Standard]
  {StartDef}{EndDef}
\newfont\<Name>{Fontname}
\newif\if<Name>
\newlength\<Name>
\newline
\newpage
\newsavebox\<Boxname>
\newtheorem{Name} [Zähler] {Titel} [Umgebung]
\nocite{Key-Liste}
\noindent
\nointerlineskip
\nolinebreak [Wert]
\nonfrenchspacing
\nonumber
\nopagebreak [Wert]
\normalbaselines
\normalmarginpar
\normalsize
\not\<Makro>
\null
\obeyspaces
\offinterlineskip
\onecolumn
\oval(Breite,Höhe) [Auswahl]
\overbrace{Gleichung}
\overline{Gleichung}
\P
\pagebreak [Wert]
\pagenumbering{Stil}
\pageref{Marke}
\pagestyle{Typ}
\par
\paragraph{Titel}
\paragraphmark{Argument}
\parbox [vPos] [Höhe] [iPos] {Breite}{Inhalt}
\PassOptionsToPackage{Optionen}{Paket}
\PassOptionsToClass{Optionen}{Paket}
\ProcessOptions*
\protect\<Makro>
\providecommand*\<Name> [N] [Standard] {Def.}
\ProvidesClass{Name} [Versionsdaten]
\ProvidesFile{Name} [Versionsdaten]
\ProvidesPackage{Name} [Versionsdaten]
\put(x,y){Objekt}
\quad
\qquad
\raggedbottom
\raggedleft \raggedright
\raisebox{Länge} [Oberlänge] [Unterl.] {Text}
\rbrack
\ref{Marke}
\refstepcounter{Zähler}
\renewcommand*\<Name> [N] [Standard] {Def.}
\repeat
\RequirePackage [Optionen] {Paket} [Datum]
\RequirePackageWithOptions{Paket} [Optionen]
\reversemarginpar
\right{Klammertyp}
\rightmark
\rlap{Text}
\rmdefault
\rmfamily
\Roman{Zähler}
\roman{Zähler}
\rq
\rule [Shift] {Breite}{Höhe}
\S
\samepage
\savebox\<Boxname> [Breite] [Pos] {Text}
\sb{Argument}
\sbox\<Boxname>{Text}
\scdefault
\scriptsize
\scshape
\scriptstyle
\scriptscriptstyle
\section * {Titel}
\sectionmark{Argument}
\selectfont
\seriesdefault
\setcounter{Zähler}{Wert}
\setlength\<Länge>{Wert}
\SetMathAlphabet\<Name>{Mathversion}
  {encoding}{family}{series}{shape}
\SetSymbolFont{Name}{Mathversion}
  {encoding}{family}{series}{shape}
\settodepth\<Länge>{Text}
\settoheight\<Länge>{Text}
\settowidth\<Länge>{Text}
\sfdefault
\sffamily
\shapedefault
\shortstack [Pos] {...\\...\\...}
\showhyphens{Wort}
\sldefault
\slash
\slshape
\sloppy
\small
\smallskip
\space
\sp
\sqrt [Wurzel] {Radikand}
\stackrel{Oben}{Normal}
\stepcounter{Zähler}
\strut
\subparagraphmark{Argument}
\subsection * {Titel}
\subsectionmark{Argument}
\subsubsection * {Titel}
\subsubsectionmark{Argument}
\suppressfloats [Position]
\symbol{Zeichennummer}
\tableofcontents
\TeX
\textbf{Text}
\textit{Text}
\textmd{Text}
\textnormal{Text}
\textrm{Text}
\textsc{Text}
\textsf{Text}
\textsl{Text}
\textstyle
\textsubscript{Argument}
\textsuperscript{Argument}
\texttt{Text}
\textup{Text}
\thanks{Text}
\the<Zähler>
\the\<Länge>
\thinspace
\thispagestyle{Seitenstil}
\tiny
\title{Text}
\today
\ttdefault
\ttfamily
\twocolumn [Überschrift]
\typein [\<Befehlsname>] {Text}
\typeout{Argument}
\underbar{Argument}
\underbrace{Gleichung}
\underline{Argument}
\updefault
\upshape
\usebox\<Boxname>
\usecounter{Zähler}
\usefont{Encoding}{Familie}{Serie}{Shape}
\usepackage [Optionen] {Name(n)}
\value{Zähler}
\vdots
\vector(dx,dy){Länge}
\verb*<Zeichen> Text <Zeichen>
\vfill
\vline
\vphantom{Text}
\vspace*{Länge}
```

```
\abovecaptionskip
\arraycolsep
\arrayrulewidth
\baselineskip
\belowcaptionskip
\bigskipamount
\columnsep
\columnseprule
\columnwidth
\dblfloatsep
\depth
\doublerulesep
\dbltextfloatsep
\emergencystretch
\evensidemargin
\fboxrule
\fboxsep
\fill
\floatsep
\fontdimen1\font
\fontdimen2\font
\fontdimen3\font
\fontdimen4\font
\fontdimen5\font
\fontdimen6\font
\fontdimen7\font
\footins
\footnotesep
\footskip
\headheight
\headsep
\height \hoffset
\intextsep
\linewidth
\marginparsep
\marginparwidth
\mathindent
\maxdimen
\medskipamount
\normalbaselineskip
\normallineskip
\normallineskiplimit
\oddsidemargin
\overfullrule
\paperheight
\paperwidth
\parindent
\parskip
\smallskipamount
\tabcolsep
\textheight
\textfloatsep
\textwidth
\topskip
\totalheight
\unitlength
\voffset
\width
```

```
\abovedisplayshortskip
\abovedisplayskip
\belowdisplayshortskip
\belowdisplayskip
\medmuskip
\thickmuskip
\thinmuskip
```

```
\itemindent
\itemsep
\labelsep
\labelwidth
\leftmargin
\listparindent
\parsep
\partopsep
\rightmargin
\topsep
```

```
\p@
\z@
\z@skip
\@tempdima
\@tempdimb
\@tempdimc
\@tempskipa
\@tempskipb
```

Zähler und Spezielles

```
part                  \@MM
chapter               \m@ne
section               \@ne
subsection            \@percentchar
subsubsection         \@vpt
paragraph             \@vipt
subparagraph          \@viipt
page                  \@viiipt
equation              \@xpt
figure                \@xipt
table                 \@xiipt
footnote              \@xivpt
mpfootnote            \@xviipt
enumi                 \@xxpt
enumii                \@xxvpt
enumiii               \csname
enumiv                \endcsname
secnumdepth           \f@size
lefthyphenmin         \hexnumber@{Zähler}
righthyphenmin        \sixt@@n
errorcontextlines     \strip@prefix{Argument}
\@tempcntb            \strip@pt\<Länge>
\@multicnt            \thr@@
\@cclvi               \tw@
\@gobble{Argument}    \two@digits{Zahl}
\@ixpt                \verbatim@font
\@m                   \@temptokena
\@M
```

Längen

1 bp Big Point (72 bp/in)	1.00374pt
1 cc Cîcero (1 cc=12 dd)	12.8401pt
1 cm Centimeter	28.45274pt
1 dd Didôt (1157 dd = 1238 pt)	1.07pt
1 em Breite von M im aktuellen Font	9.0pt
1 ex Höhe von x im aktuellen Font	4.22095pt
1 in Inch (korrekt 72.27 pt)	72.26999pt
1 mm Millimeter	2.84526pt
1 mu math unit (1 mu=1/18em)	0.05554pt
1 pc Pica (12 pt/pc)	12.0pt
1 pt (TeX-)Point (1/72.27 Inch)	
1 px Bildpunkt (pdfTeX)	
1 sp Scaled Point (65536 sp/pt)	0.00002pt

Dokumentenklassen (Auswahl)

IEEEconf	exam	memoir	ua-thesis
IEEEtran	extarticle	muthesis	ucthesis
article	extbook	powerdot	uiucthesis
beamer	extreport	report	umich-thesis
book	lashcard	sciposter	uwthesis
combine	g-brief2	scrartcl	york-thesis
dinbrief	kluwer	scrbook	
ebsthesis	leaflet	scrreprt	
elsart	ltugboat	scrlttr2	

20 Jahre dante e. V.

Stammtische

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

Aachen
Torsten Bronger `bronger@physik.rwth-aachen.de`
Gaststätte Knossos, Templergraben 28
Zweiter Donnerstag im Monat, 19.00 Uhr

Berlin
Rolf Niepraschk
Tel.: 0 30/3 48 13 16
`Rolf.Niepraschk@gmx.de`
Gaststätte Bärenschenke
Friedrichstraße 124
10117 Berlin Mitte
Zweiter Donnerstag im Monat, 19.00 Uhr

Bremen
Winfried Neugebauer
Tel.: 04 21-8 28 65 14
`tex@wphn.de`
Wechselnder Ort
Erster Donnerstag im Monat, 18.30 Uhr

Darmstadt
Karlheinz Geyer
`geyerk.fv.tu@nds.tu-darmstadt.de`
Restaurant Poseidon, Rheinstraße 41
64283 Darmstadt
Erster Freitag im Monat, ab 19.30 Uhr

Dresden
Carsten Vogel
`lego@wh10.tu-dresden.de`
Studentenwohnheim, Borsbergstraße 34,
Dresden, Ortsteil Striesen
ca. alle 8 Wochen, Mittwoch, 19.00 Uhr

Düsseldorf
Georg Verweyen
`Georg.Verweyen@web.de`
Bistro/Café Zicke
Böckerstr. 5 a (Ecke Bergerallee)
40213 Düsseldorf
Zweiter Mittwoch in ungeraden Monaten, 20.00 Uhr

Erlangen
Walter Schmidt, Peter Seitz
`w.a.schmidt@gmx.net`
Gaststätte »Deutsches Haus«
Luitpoldstraße 25
3. Dienstag im Monat, 19.00 Uhr

Freiburg
Heiko Oberdiek
Tel.: 07 61/4 34 05
`oberdiek@uni-freiburg.de`
Wechselnder Ort
Dritter Donnerstag im Monat, 19.30 Uhr

Hamburg
Lothar Fröhling
`lothar@thefroehlings.de`
Zum Schwarzenberg
Schwarzenbergstr. 80 – 21073 HH
Letzter Dienstag im Monat, 19.30 Uhr

Hannover
Mark Heisterkamp
`heisterkamp@rrzn.uni-hannover.de`
Seminarraum RRZN
Schloßwender Straße 5
Zweiter Donnerstag im Monat, 18.30 Uhr

Heidelberg
Luzia Dietsche
Tel.: 0 62 21/54 45 27
`luzia.dietsche@urz.uni-heidelberg.de`
»Restaurant Tomato, der Turm«
Alte Glockengießerei 9
Letzter Mittwoch im Monat, 19.30 Uhr

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

Köln
Helmut Siegert
Institut für Kristallographie
Zülpicher Straße 49b
Letzter Mittwoch im Monat, 19.30 Uhr

München
Uwe Siart
uwe.siart@tum.de
http://www.siart.de/typografie/stammtisch.xhtml
Gaststätte »Marktwirt«
Heiliggeiststr. 2
Erste Woche des Monats an wechselnden Tagen, 19.00 Uhr

Stuttgart
Bernd Raichle
bernd.raichle@gmx.de
Bar e Ristorante »Valle«
Geschwister-Scholl-Str. 3
Zweiter Dienstag im Monat, 19.30 Uhr

Trier
Martin Sievers
stammtisch-trier@texberatung.de
Fetzenkneipe (Haus Fetzenreich)
Sichelstraße 36 (beim Sieh-Um-Dich)
Dritter Montag des Monats, 20.15 Uhr

Ulm
Adelheid Grob
adelan@heidi.in-ulm.de
http://latex.in-ulm.de
Gaststätte »Peppers Ulm«
Deinselsgasse 8
Erster Donnerstag im Monat, 19.30 Uhr

Wuppertal
Andreas Schrell
Tel.: 02193/53 10 93
as@schrell.de
Restaurant Croatia »Haus Johannisberg«
Südstraße 10
an der Schwimmoper Wuppertal-Elberfeld
Zweiter Donnerstag im Monat, 19.30 Uhr

Würzburg
Bastian Hepp
LaTeX@sning.de
nach Vereinbarung

Adressen

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

Tel.: 0 62 21/2 97 66 (Mo., Mi.–Fr., 10.00–12.00 Uhr)
Fax: 0 62 21/16 79 06
E-Mail: dante@dante.de

Konto: Volksbank Rhein-Neckar eG
BLZ 670 900 00
Kontonummer 2 310 007
IBAN DE67 6709 0000 0002 3100 07
SWIFT-BIC GENODE61MA2

Präsidium

Präsident:	Klaus Höppner	president@dante.de
Vizepräsident:	Volker RW Schaa	vice-president@dante.de
Schatzmeister:	Tobias Sterzl	treasurer@dante.de
Schriftführer:	Manfred Lotz	secretary@dante.de
Beisitzer:	Günter Partosch	
	Bernd Raichle	
	Herbert Voß	

Server

ftp: ftp.dante.de
WWW: http://www.dante.de/

Autoren/Organisatoren

Jürgen Fenn [73]
Friedensallee 174/20
63263 Neu-Isenburg
juergen.fenn@gmx.de

Adelheid Grob [59]
Sedanstr. 67
89077 Ulm
latex@heidi.in-ulm.de

Klaus Höppner [4]
siehe Seite 82

Joachim Lammarsch [5]
Joachim.Lammarsch@urz.uni-heidelberg.de

Gerd Neugebauer [66]
Im Lerchelsböhl 5
64521 Groß-Gerau
gene@gerd-neugebauer.de

Michael Niedermair [26]
Am Malerwinkel 16
85778 Haimhausen
m.g.n@gmx.de

Philipp H. Poll [26]
PhilippPoll@web.de

Volker RW Schaa [4]
siehe Seite 82

Hàn Thế Thành [21]
hanthethanh@gmail.com

Herbert Voß [3, 76]
siehe Seite 82

Dominique Wagenführ [50]
dwagenfuehr@freiesmagazin.de

Uwe Ziegenhagen [35]
Berlin
uwe@ziehenhagen.info

Die TEXnische Komödie

21. Jahrgang Heft 1/2009 Februar 2009