Die TEXnische Komödie

dante
Deutschsprachige
Anwendervereinigung TEX e.V.

29. Jahrgang Heft 1/2017 Februar 2017

1/2017

Impressum

»Die TEXnische Komödie« ist die Mitgliedszeitschrift von DANTE e.V. Der Bezugspreis ist im Mitgliedsbeitrag enthalten. Namentlich gekennzeichnete Beiträge geben die Meinung der Autoren wieder.

Beiträge sollten in Standard-LATEX-Quellcode unter Verwendung der Dokumentenklasse `dtk` erstellt und per E-Mail oder Datenträger (CD/DVD) an untenstehende Adresse der Redaktion geschickt werden. Sind spezielle Makros, LATEX-Pakete oder Schriften notwendig, so müssen auch diese komplett mitgeliefert werden. Außerdem müssen sie auf Anfrage Interessierten zugänglich gemacht werden. Weitere Informationen für Autoren findet man auf der Projektseite `http://projekte.dante.de/DTK/AutorInfo` von DANTE e.V.

Diese Ausgabe wurde mit `LuaTeX, Version 1.0.2 (TeX Live 2017/dev)` erstellt. Als Standard-Schriften kamen Libertinus Serif, Libertinus Sans, Anonymous Pro und Libertinus Math zum Einsatz.

Erscheinungsweise: vierteljährlich
Erscheinungsort: Heidelberg
Auflage: 2400
Herausgeber: DANTE, Deutschsprachige Anwendervereinigung TEX e.V.
Postfach 10 18 40
69008 Heidelberg

E-Mail: `info@dante.de` (DANTE e.V.)
`dtkred@dante.de` (Redaktion)

Druck: Konrad Triltsch Print und digitale Medien GmbH
Johannes-Gutenberg-Str. 1–3, 97199 Ochsenfurt-Hohestadt

Redaktion: Herbert Voß (verantwortlicher Redakteur)
Mitarbeit: Gert Ingold, Eberhard Lisse, Rolf Niepraschk, Günter Partosch, Christine Römer, Volker RW Schaa, Martin Sievers

Redaktionsschluss für Heft 2/2017: 15. April 2017

ISSN 1434-5897

Editorial

Liebe Leserinnen und liebe Leser,

die dritte Ausgabe im Jahr 2011 wurde erstmalig mit LuaLaTeX gesetzt. Der Grund waren zwei Artikel, die spezifische Inhalte zu LuaTeX zeigen. Da dieser Versuch erfolgversprechend war, blieben wir einfach bei dieser neuen Variante von TeX und sind nun schon im siebten »Luajahr« angekommen. Jetzt könnte ich ganz euphorisch sein und erwähnen, dass endlich LuaTeX offiziell mit der Version 1.0 aus dem Betastadium in das »reale Compilerleben« entlassen wurde. Im Gegensatz zu den Äußerungen, die man im Netz lesen konnte, hat sich aber fast nichts geändert; die Version 1.0 unterscheidet sich faktisch nicht von der Vorgängerversion 0.95. Ohne weitere Vereinbarung im Dokument war bislang die Computer Modern die Standardschrift. Nun ist es die Latin Modern, wobei man sich das Laden des Paketes `fontspec` ersparen kann, wenn man diese Schrift auch benutzen will. Die Anwendung von LuaLaTeX ist für uns mittlerweile zur Routine geworden. Daneben haben wir eine Infrastruktur geschaffen, die es uns ermöglicht, Beispiele während des laufenden Übersetzungsprozesses auch mit ConTeXt oder jedem anderen Compiler laufen zu lassen.

Als Autor muss man sich jedoch um diese formalen Dinge im Prinzip nicht kümmern. Im Zweifel kann man ein Manuskript mit der Klasse `article` erstellen und mit `pdflatex` laufen lassen oder alternativ komplett mit ConTeXt; eine Konvertierung für die Anwendung der Dokumentenklasse `dtk` und einem anschließenden Lauf mit LuaLaTeX ist uns bislang noch immer gelungen.

Ich wünsche Ihnen wie immer viel Spaß beim Lesen und verbleibe

mit TeXnischen Grüßen

Ihr Herbert Voß

Hinter der Bühne

Vereinsinternes

Grußwort

Liebe Mitglieder,

das letzte Grußwort liegt noch gar nicht so lange zurück und doch kann ich schon wieder über einige Dinge berichten.

Zunächst möchte ich mich für die zahlreichen Vorschläge für den Ehrenpreisträger 2017 bedanken. Es waren allesamt geeignete Kandidaten, so dass der Vorstand bei seiner Sitzung am 7. Januar die »Qual der Wahl« hatte. Die Entscheidung, wer Urkunde und Preisgeld erhält, wird wie angekündigt im Rahmen der Mitgliederversammlung während der Frühjahrstagung bekannt gegeben.

Nicht nur deswegen möchte ich Sie nochmals ganz herzlich nach Zeuthen einladen. Mittlerweile stehen einige Vortragsthemen fest und wir sind bemüht, das Programm möglichst bald »festzuzurren«. Wie immer hilft uns Ihre baldige Anmeldung, um besser planen zu können. Bitte beachten Sie insbesondere die Hotelkontingente, die bis Mitte bzw. Ende Februar verfügbar sind.

Besonders freue ich mich, dass diesmal voraussichtlich einige externe Gäste das Programm bereichern. So wird z. B. Klaas Posselt von der PDF Association einen Vortrag zum Thema »Barrierefreiheit und PDF/UA« halten.

In diesem Zusammenhang weise ich gerne auf die Möglichkeit hin, dass Mitglieder von DANTE e.V. auch in diesem Jahr die »PDF Days Europe« zu vergünstigten Preisen besuchen können.

2017 werden wir zudem als Verein neben den beiden eigenen Tagungen wieder auf einigen externen Veranstaltungen als Aussteller vertreten sein. Die Linux- bzw. Open-Source-Tage in Chemnitz, Oberhausen und St. Augustin gehören dabei seit Jahren zum festen Repertoire.

In diesem Jahr kommen mit dem Informatiktag NRW und mit dem TEX-Zelt auf der SHA 2017 (noch in Planung) zwei Veranstaltungen hinzu, die jeweils andere und ganz unterschiedliche Zielgruppen ansprechen. Beide Wege sind aus meiner Sicht sehr wichtig, um links und rechts unserer »ausgetrampelten Pfade« weitere Interessenten für TEX zu gewinnen.

Ich möchte mich an dieser Stelle bei allen Freiwilligen bedanken, die durch Ihren Standdienst o. ä. zum guten Gelingen beitragen und DANTE e.V. in der Öffentlichkeit repräsentieren.

Leider brachte das neue Jahr auch traurige Nachrichten. So müsste ich erfahren, dass unser Gründungsmitglied Klaus Braune über die Weihnachtstage verstorben ist. Vieles, was er bewegt hat, geschah vor meiner aktiven Zeit bei DANTE e.V. Ich selbst habe ihn aber noch im Herbst 2014 als äußerst engagierten Organisator unserer Herbsttagung in Karlsruhe erlebt.

Neben Klaus Braune verstarben im vergangenen Jahr neun weitere Mitglieder unseres Vereins: Christian Brücker, Hans-Günter Schmidt, Harald Dunker, Volkmar Weise, Dieter Schroeder, Olav Wilde, Berthold Beyer, Helmut Siegert sowie Herbert Möller. Allen Trauernden gilt unsere aufrichtige Anteilnahme.

Zum Schluss noch etwas aus meinem Alltag mit TEX: Schon das ein oder andere Mal habe ich im Grußwort oder auch auf Tagungen über meine Erfahrungen zum Einsatz von TEX in den Geisteswissenschaften berichtet. Über die vereinsinterne Mailingliste hat sich eine Gruppe von Anwendern gefunden, die ein Handbuch genau zu diesem Thema erstellen wollen. Auf Github (`https://github.com/thomas-hilarius-meyer/LaTeX-fuer-Geisteswissenschaftler`) entsteht gerade etwas, das am Schluss hoffentlich auch als Buch erscheint und weiteren Personen TEX und seine vielfältigen Einsatzmöglichkeiten näher bringt.

Ein Problem in meinem unmittelbaren Anwendungsbereich ist und bleibt die Verarbeitung von XML-Dokumenten. Es gibt sehr gute Ansätze in ConTEXt, doch für Nutzer von LATEX bleiben nur Umwege. Nun bin ich im Rahmen meiner Arbeit (mal wieder) über PassiveTEX (`http://projects.oucs.ox.ac.uk/passivetex`) gestolpert, ein System, das der im letzten Jahr leider verstorbene Sebastian Rahtz vor vielen Jahren für die Verarbeitung von XML entwickelt hat. Er hatte schon früh erkannt, dass man XML und TEX miteinander verbinden sollte.

Auch wenn es auch von anderer Seite immer wieder Überlegungen und erste Umsetzungen gab, so harrt dieser »Verbindungsprozess« doch weiterhin seiner Vollendung. Doch wer weiß: Vielleicht geht es 2017 auch in diesem Bereich von TEX ein gutes Stück voran.

In diesem Sinne wünsche ich Ihnen viel Vergnügen bei der weiteren Lektüre.

Herzlichst Ihr/Euer
Martin Sievers

Das 10. ConTEXt-Meeting in Kalenberg, Niederlande, 25. 9.–1. 10. 2016

Henning Hraban Ramm

Happy Birthday, ConTEXt!

Beim diesjährigen Treffen der ConTEXt-Freunde kamen mehrere runde Jubiläen zusammen: Vor 10 Jahren wurde die »ConTEXt Group« gegründet, seit 20 Jahren gibt es ConTEXt und seit 30 Jahren Pragma ADE, die Firma, die ConTEXt maßgeblich entwickelt hat. Zusätzlich wird seit zehn Jahren an LuaTEX gearbeitet, das während dieser Veranstaltung in der stabilen Version 1.0 freigegeben wurde. Es gab also mehrere Anlässe zum Feiern!

Ankommen

Das Dorf Kalenberg liegt im Nationalpark Weerribben und war bis in die 1950er Jahre nur per Boot zu erreichen. Ich wurde freundlicherweise von Michael und Susanna Guravage vom Bahnhof Zwolle abgeholt und nach einer guten Stunde auf dem ungepolsterten Rücksitz einer Ente und einem kleinen Umweg, weil das Navi uns eine Fußgängerbrücke über einen Kanal nahelegte, kamen wir an unserer Unterkunft »Het Doevehuis« (`http://www.hetdoevehuis.nl/`) an, wo ein Teil der internationalen Teilnehmer schon eifrig am Diskutieren und Installieren war.

Die Bewohner von Kalenberg und Umgebung lebten früher hauptsächlich vom Torfabbau und dem Reethandel – der Weerribben (*Weer* ist Wasser und *Ribben* ein nasser Torfplacken) ist eine Moorlandschaft mit großen Schilfflächen, durchzogen von Kanälen und kleinen Seen.

Während der Reethandel immer noch eine Rolle spielt, steht der Torf heute unter Naturschutz. Fast jedes Haus hat seinen eigenen kleinen Hafen, der Wasserpegel liegt nur kurz unter der Rasenkante. »Het Doevehuis« (das Taubenhaus) gehörte ursprünglich einer Familie Taube, wurde dann von einer Studentenvereinigung betrieben und ist heute ein Restaurant mit Gruppenunterkunft auf dem Standard alter Jugendherbergen. Unsere Verpflegung im Restaurant war reichhaltig und lecker (für meine Begriffe allerdings etwas vitaminarm) mit einigen niederländischen Eigenheiten.

Hören, Sehen und Staunen

Im Eröffnungsvortrag am Montagmorgen stellte Ton Otten den automatisierten Workflow vor, mit dem bei Pragma ADE Mathematikbücher gesetzt werden. Dass Pragma von Schulbüchern lebt, war mir bekannt, aber über das »Wie« hatte ich mir vorher keine großen Gedanken gemacht. Da Redaktionssysteme und automatisierte Publishing-Workflows zu meinen besonderen Interessensgebieten gehören, war das ein interessanter und inspirierender Einblick.

Die meisten Referate über die ganze Woche wurden von Hans Hagen persönlich bestritten, von der Vorstellung neuer Möglichkeiten und neuer Dokumentationen über Details, die der einfache Anwender gar nicht wissen möchte, bis zur Erinnerung an uralte Funktionen, die selten genutzt werden oder die in jüngster Zeit aufpoliert wurden:

- Spationierung im Formelsatz.
- Probleme der Interpretation mathematischer Eingaben (besonders von ascii math).
- Anreicherung von Schriften mit zusätzlichen Funktionen (Emulation der OpenType-Tabellen in LuaTEX).
- Unterstützung von Farbinformationen in Schriften, die es seit der Aufblähung des Unicode-Standards für Emojis gibt (zusammen mit Arthur Reutenauer).
- Die neue OpenType-Version des Kuh-Fonts[1] enthält auch Farbinformationen (zusammen mit Taco Hoekwater).

[1] Die Schrift ist aus urheberrechtlichen Gründen nur in der ConTEXt-Distribution enthalten, nicht in TEXLive.

- Farben: Neu ist vor allem das Handbuch zum Thema.
- Zeilenausgleich mit Schlangen: Von Mojca Miklavec mit einem xkcd-Cartoon (`http://xkcd.com/1676/`) herausgefordert, fand Hans heraus, wie man Zeilen mit grafischen Elementen füllen kann, obwohl das mit TEX eigentlich nicht möglich ist. Dazu war es auch nötig, die Verwendung von Linien durch TEX durch zusätzliche Hooks in LuaTEX konfigurierbar zu machen. Das Ersetzen von Linien durch Font-Glyphen war auch für die Umsetzung von Wurzeln mit dem Kuh-Font nötig (zusammen mit Taco Hoekwater).
- Neuigkeiten in MetaPost: Es gab einige Vereinfachungen, besonders im Zusammenhang mit Farben (zusammen mit Luigi Scarso).
- Status von LuaTEX: Geschichte der Entwicklung und die Entscheidung, die aktuelle Version als 1.00 freizugeben (zusammen mit Luigi Scarso).
- Skripte: Welche Werkzeuge finden sich in der Distribution und was kann man damit tun? Während alle neueren Skripte in Lua geschrieben sind, werden auch die veralteten Ruby- und Perl-Skripte noch mitgeliefert.
- Handbücher (`http://www.pragma-ade.com/overview.htm`): Im letzten Jahr wurden mehrere Kapitel neu geschrieben und ältere aktualisiert. Weitere sind in Arbeit.
- Mehrspaltiger Satz: Der vierte Spaltensatz-Modus »mixedcolumns« ist bei Pragma bereits seit zwei Jahren im Einsatz, aber jetzt erst veröffentlicht. Er kann den ältesten Spaltensatzmechanismus direkt ersetzen.
- Erweiterungen durch externe Hilfsprogrammen: z. B. Datenbank-Anbindung und Bildkonvertierungen.
- Workflows: Hinweise und hilfreiche Funktionen zur Verwaltung, zum Debugging und zur effizienten Einrichtung von Projekten.

Zusätzlich zu den weitgehend von Hans Hagen verfassten Handbüchern und Anleitungen verfasst Alan Braslau ein englischsprachiges ConTEXt-Handbuch für Anfänger aus den Naturwissenschaften. Es ist noch nicht fertig, hat aber schon einen Verlag in den USA.

Philipp Gesang schilderte die Portierung des OpenType-Fontloaders (`https://github.com/lualatex/luaotfload`) aus ConTEXt, den er für andere Projekte verfügbar gemacht hat – er wird nicht nur mit LuaLATEX, sondern beispielsweise auch mit Gregorio eingesetzt (`http://gregorio-project.github.io/`).

Tobias Berndt referierte über den Fraktursatz, seine Geschichte und seine Regeln. Es entspann sich eine lebhafte Diskussion, wie viel davon automatisierbar wäre, beispielsweise durch Ligaturtabellen in OpenType-Schriften.

Tomáš Hála präsentierte die Arbeit seines Studenten Marek Treťák zur Heraldik von Flaggen (Vexillographie). Besonders beeindruckend waren die detailreich in MetaPost programmierten Nationalwappen.

Passend zum Geburtstagsthema der Veranstaltung »Piece of Cake« organisierte Taco Hoekwater zwei Kuchen-Workshops, bei denen jeweils zwei Teilnehmer zusammen nach ausführlicher Anleitung einen individuellen Kuchen backen sollten. Die entstandenen Backwaren bereicherten unsere Kaffeepausen, ergänzt durch typisch holländische Kekse, die uns Ton Otten in einem Kurzreferat erklärte.

Als Geschenk an die Teilnehmer gab es passenderweise einen Kuchenteller mit einem von Duane Bibby gezeichneten Motiv. Zusätzlich gab Taco einen Einblick in die Geschichte der Zeichensetzung und präsentierte seine neue Version der Webanwendung zur Verwaltung der ConTEXt-Module: `http://modules.contextgarden.net` wurde von einer Ruby-Anwendung bedient, die seit fast zwei Jahren nicht mehr benutzbar war. Die neue Version (geschrieben in Perl, erstaunlicherweise nicht Lua) kann Modulversionen direkt von den Websites oder aus den Sourcecode-Repositories der Autoren holen. Weiterhin führte er vor, welche Informationen ConTEXt in der `tuc`-Datei ablegt.

Harald König stellte den aktuellen Stand seines Projektes dar, die Layout-Dateien des Cewe-Fotobuch-Designers in ConTEXt zu konvertieren. Ausgangspunkt war die Nachfrage nach einer PDF-Version eines Fotobuchs und die Entdeckung, dass das Programm xml-Daten erzeugt.

Arthur Reutenauer erklärte den Aufbau von TEX-Trennmustern und sein Projekt »Hydra«[2], mit dem er die Trennmuster für mehrere Sprachen vereinfacht und verbessert. Mojca Miklavec erstellte eine Buildbot-Infrastruktur für LuaTEX, mit der die Binärprogramme für die verschiedensten Architekturen bei jeder Codeanpassung automatisch erzeugt werden (Continuous Integration). Damit können Kompilierungsprobleme sofort erkannt werden (nicht erst bei der Zusammenstellung von TEXLive) und abenteuerlustige Anwender können frühzeitig auf neue Versionen zugreifen. Zu letzteren werden alle ConTEXt-Anwender gezählt, die mit

[2] `https://github.com/hyphenation/hydra`

der »minimals«-Distribution arbeiten, welche regelmäßig die neuesten Versionen mitliefert.

Wolfgang Schuster verbrachte mehrere Monate damit, alle Befehlsdefinitionen aus den ConTEXt-Quellen von Hand in Notizbücher einzutragen. Beim bestaunten Exemplar beneideten ihn mehrere Teilnehmer um seine klare Handschrift.

Davon ausgehend schrieb er Interface-Dokumentationen als xml-Dateien, die jetzt mit der Distribution ausgeliefert werden[3] und die Befehlsreferenz im Wiki ersetzen sollen. Ich habe noch während des Treffens den »Proof of Concept« eines Befehlsbrowsers in JavaScript geschrieben, der sich aus diesen Dateien speist (`https://github.com/fiee/contextref`).

Außerdem hatte ich das Vergnügen, das Notensatzsystem GNU-LiliPond und die Einbindung von LilyPond-Quellcode in ConTEXt-Projekte vorzustellen. Üblicher ist die Einbindung über den LATEX-Präprozessor `lilypond-book`. Bemerkenswert sind auch die jüngsten Arbeiten von Urs Liska und anderen, die den Satz kritischer Ausgaben von Musikwerken erleichtern.

Land unter Wasser

Während wir all den interessanten Referaten trotz schönsten Wetters im abgedunkelten Raum gelauscht hatten, fand unser Ausflug bei strömendem Regen statt. Allzu nass wurden wir dennoch nicht, denn ein Bus brachte uns nach Lelystad am Ufer des Ijsselmeers, wo wir im Neulandmuseum (`http://www.nieuwlanderfgoed.nl/`) eine Einführung in die Geschichte der Niederlande und des Kampfes gegen das Wasser erhielten.

[3] Als PDF auch unter `http://pragma-ade.com/general/qrcs/setup-en.pdf`, die XML-Quellen liegen in der Distribution unter `texmf-context/tex/context/interface/`.

Während bereits im 17. Jahrhundert Wasserflächen trockengelegt wurden, begannen die großen Polderprojekte erst Anfang des 20. Jahrhunderts, als Dampfmaschinen für die Erdarbeiten und Wasserpumpen zur Verfügung standen. In den 1990er Jahren wurde dieses moderne Weltwunder für abgeschlossen erklärt.

Heute gilt die Sorge mehr den zunehmenden Wassermengen, welche die Flüsse ins Land bringen – durch Klimaveränderungen und die Verbauung der flutbaren Pufferflächen kommen starke Regenfälle und Schmelzwasser direkt bis zur Mündung durch und verursachen Überschwemmungen der tief liegenden Landflächen.

Das Museum begeisterte uns mit seinem abwechslungsreichen, modernen Ausstellungskonzept und befriedigte unseren Spieltrieb an den Experimentierstationen. Meine Kinder wären dort sicherlich wochenlang beschäftigt gewesen, auch wenn ich mich frage, wie ich sie zur Einhaltung der Hausregeln (Schuhe und Kleidung anbehalten, nicht rennen und nicht schreien) bringen könnte.

Am Freitag wurden wir schließlich noch direkt vor der Haustür zu einer Bootsfahrt durch den Weerribben abgeholt. Von einem fast unhörbaren Elektromotor angetrieben, glitten wir durch die großen und kleinen Kanäle im Moor. An einem der Anlege- und Zeltplätze für Kanutouristen stiegen wir aus, um die Elastizität des schwimmenden Bodens auszuprobieren.

Unser Schiffer hatte eine Menge Geschichten über diese Kulturlandschaft zu erzählen, z. B. über die Wiederansiedlung der ausgerotteten Otter: Es gab dagegen Proteste der Fischer, die um ihren Fang fürchteten. Seit der Auswilderung von Ottern stieg allerdings die Fischpopulation stark an, weil der Otter vor allem die Raubfische dezimierte.

Ausblick

In den letzten Jahren ist viel geschehen in der ConTEXt-Welt: Schnittstellen wurden stabil, die Dokumentation deckt immer größere Teile der Möglichkeiten ab, und die Infrastruktur wurde modernisiert. Gleichzeitig sind Pragma ADE und die Benutzergemeinde aktiv dabei, die Benutzung zu vereinfachen und die Nutzungsmöglichkeiten zu verbreitern. LuaTEX und ConTEXt sind also gut für die Zukunft aufgestellt.

Das nächste ConTEXt–Meeting (`http://meeting.contextgarden.net/`) wird vom 11. bis 17. September 2017 in Deutschland stattfinden, und zwar im Seminarhaus »Maibacher Schweiz« (`http://maibacher-schweiz.de/`) bei Butzbach, ca. 40 km nördlich von Frankfurt am Main. Natürlich freuen wir uns über neue Gesichter und alte Hasen (Löwen?)!

Bretter, die die Welt bedeuten

GUST e-foundry Font Projects

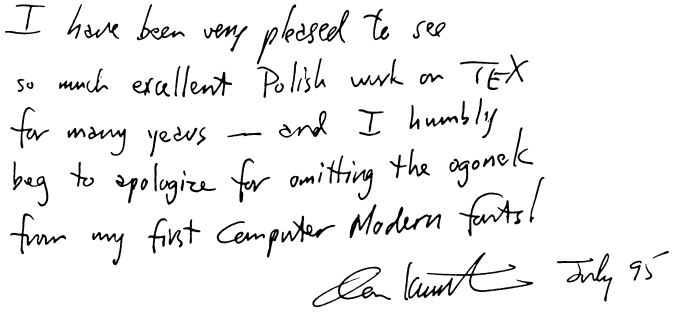
Bogusław Jackowski, Piotr Strzelczyk, Piotr Pianowski

What is a document? It is a sequence of rectangles containing a collection of graphic elements. What is a font? It is a sequence of rectangles containing a collection of graphic elements.

— Marek Ryćko

Introduction

The Polish TEX Users Group (GUST) has paid attention to the issue of the fonts since the beginning of its existence. In a way, it was a must, because the repertoire of the diacritical characters of the *Computer Modern* family of fonts (CM), »canonical« TEX family defined as METAFONT programs (see [5]), turned out to be insufficient for the Polish language. The efforts of the GUST font team (GUST e-foundry), led by Bogusław Jackowski, were kindly acknowledged by Professor Donald E. Knuth:



Obviously, our first fonts were PK bitmap fonts programmed using METAFONT. Alas, the TEX/METAFONT bitmap font format never became the world-wide standard. Therefore, the next step were fonts in the PostScript Type 1 format which fairly soon became obsolescent and were replaced by the OpenType format (OTF, a joint enterprise of Microsoft and Adobe, 1996, see [31]) which is actually a common container for the Adobe PostScript Type 1 and Microsoft TrueType (TTF) formats. In

2007, Microsoft extended the OTF standard with the capability of typesetting math formulas, largely based on ideas developed for TEX, and implemented it in MS Office. Soon, TEX engines were adapted to process such math OTF fonts. Therefore our recent fonts are released in the OpenType format, which also makes them easily usable outside the TEX realm.

So far, no OTF successor is in sight, which is both good and bad (cf. page 48).

We published our partial results successively as the work progressed. This paper provides an overall summary of our work. It describes the collections of fonts prepared by the GUST e-foundry, deals with some technical issues related to the generation of fonts and their structure and puts forward a few proposals concerning future works.

This is not an overly strict report, but rather a story about our technical work on fonts, illustrated by representative examples which, we hope, show the essence of the matter. In order to keep our narration smooth, we decided not to use formal captions with explanations to figures and tables (only numbers of figures and tables are given). The relevant detailed descriptions always appear in the main text.

Historical background

PostScript and TEX are genetically related: their common ancestor is the ingenious idea of a programming language for the description of documents understood as a sequence of rectangular pages filled with letters and graphics. Both projects were devised nearly at the same time — at the turn of the 1970s to the 1980s.[1] And both are still alive and well, proving that the idea behind both projects was indeed brilliant.

From our perspective, the most important thing in common, and at the same time a key distinctive element, was the different handling of fonts and graphics; in other words, both systems clearly distinguished illustrations from fonts. That approach was justified by the computer technology at that time.

For both TEX and PostScript, fonts were external entities, both used metric files plus files defining glyph shapes, both defined contours as Bézier splines (planar polynomials of the 3rd degree), and for both fonts were to be prepared separately with dedicated font programs, prior to creating documents.

And this exhausts the list of similarities.

TEX worked with binary metric files, TFM; its output, a device independent file (DVI), was processed by so-called drivers which made use of the »proper« fonts, that is, the relevant bitmap collections, and produced output that could be sent to a printer or to a screen. The bitmaps were prepared independently with the METAFONT

[1] Formally, TEX was released a little earlier—TEX in 1982, PostScript in 1984, both with earlier work.

program(s) which interpreted scripts written in the METAFONT language and generated TFM and bitmap files.

In METAFONT, the shapes of glyphs are defined as Bézier curves, stroked with a »pen« and/or filled.

Basic PostScript fonts (i.e., Type 1; see [28]) employ contours defined as non-intersecting closed Bézier outlines which can only be filled.[2] The »filled outline« paradigm relates also to the Microsoft TTF format and, thereby, to the OTF format.

PostScript Type 1 fonts are usually (but not necessarily) accompanied by corresponding ASCII metric files (AFM), not used by the interpreters of the PostScript language. In the Microsoft Windows operating system, making an already complex situation even more complex, binary printer font metric files (PFM) were introduced for Windows drivers that used PostScript Type 1 fonts.

For a long time, only commercial programs for generating PostScript Type 1 fonts were available. Only in 2001, George Williams released his remarkable FontForge program (initially dubbed PfaEdit; [25]). FontForge can generate outline fonts in many formats, including PostScript Type 1 and OTF.

PostScript was promptly (and rightly) hailed as *the* standard for printers and, more importantly, for phototypesetters, therefore a driver converting DVI files to PostScript became necessary. Fortunately for TEXies, PostScript is equipped also with Type 3 fonts; glyphs in Type 3 fonts can be represented by nearly arbitrary graphic objects, in particular, by bitmap. Therefore the making of a PostScript driver for converting DVI files to PostScript was possible already in 1986, when Dvips, the first and still most popular driver was released by Tomas Rokicki. (It's a pity that the idea of Type 3 fonts was not supported and developed by Adobe.)

There were a few unsuccessful attempts to convert the basic TEX font collection, CM, to the PostScript Type 1 format automatically, thus preserving the parameterization. The main hindrance was the excessive usage of stroked (both painted and erased) elements in the CM font programs, while, as was mentioned previously, the PostScript Type 1 and OTF formats accept only filled shapes.

The »filled outline« paradigm was a convenient optimization at the beginning of the computer typesetting era, when, for example, the generation of the complete collection of bitmaps for the CM fonts at a resolution, say, of 240 dots per inch (typical for dot matrix printers) took a few days. Nowadays, the paradigm still

[2] The PostScript Type 1 documentation [28], p. 34, mentions the possibility of stroking: *a Type 1 font program can also be stroked along its outline when the user changes the PaintType entry in the font dictionary to 2. In this case, overlapping subpaths will be visible in the output; this yields undesirable visual results in outlined characters.* In practice, this possibility is not used.

thrives by virtue of tradition: there is an abundance of such fonts and, and what is worse, all operating systems support only this kind of font.

First steps

Taking the above into account, we made up our minds to design our own programmable system for generating fonts in »world-compatible« formats.

Our tools

Our primary tool was METAPOST [4], a successor of METAFONT, which promised well as a tool for making PostScript Type 1 fonts due to its native PostScript output. We called our METAPOST-based package MetaType 1 [13]. It was instantiated as a set of scripts using, besides METAPOST, T1utils, that is, Lee Hetherington's (dis)assembler for PostScript Type 1 fonts (cf. [3, 4]). A few scripts written in Gawk and Perl were also employed.

On the one hand, such a simple approach turned out to be insufficient for generating OTF fonts, in particular OTF math fonts. On the other hand, it turned out to be flexible enough to include an extra external step for making OTF fonts. For text fonts, we employed the *Adobe Font Development Kit for OpenType* (AFDKO [26]); for math fonts, the FontForge library governed by Python scripts [15, 25]. In the future, we want to replace AFDKO by a FontForge+Python utility (cf. page 42).

A set of METAPOST macros in the MetaType 1 package defines two important procedures, essential for generating non-intersecting outlines and heavily used in our font programs: finding a common outline for overlapping figures, known also as removing overlaps, and finding the outline of a pen stroke, known as expanding strokes or finding the pen envelope.

Another important feature, hinting, is implemented, but, in the end, we decided to avoid manual hinting, since it is difficult to control and yields mediocre results. METAFONT has no notion of hinting—the METAFONT language simply offers rounding. Moreover, the language for describing outlines in PostScript Type 1 fonts cannot express even as trivial a mathematical operation as rounding.

For low-resolution devices, controlled rounding is crucial—hence the idea of »hinting«, that is, controlled rounding. Alas, hinting algorithms remain undisclosed, especially with regard to commercial typesetting devices such as phototypesetters. One can presume, however, that low-resolution devices are bound to disappear sooner rather than later: the resolution of display devices has reached almost 600 dpi and 1200 dpi (and more) for printers is nowadays nothing special. Therefore,

running with the hare and hunting with the hounds, we decided to hint our recently released OTF fonts automatically with FontForge.

Trying our tools out

We tested our newborn MetaType 1 engine against a simple example, namely, Donald E. Knuth's `logo` font [17]: the sources, originally written in the METAFONT language, were adapted to MetaType 1's requirements. The distributed package contains both MetaType1 sources and the resulting PostScript Type1 files for the `logo` font [13].

The test proved the usefulness of the approach; hence, in 1999, we started a larger project: the programming of the long-established Polish typeface *Antykwa Półtawskiego* as a parameterized font. The preliminary family of fonts was released in 2000. Ten years later, prompted by the TEX community, we released the enhanced version of *Antykwa Półtawskiego* with the relevant OTF files [7]. In parallel, Janusz M. Nowacki used MetaType1 to generate several replicas of Polish fonts, namely, *Antykwa Toruńska*, *Kurier*, *Iwona*, and *Cyklop* [22].

Latin Modern collection of fonts

Recall that the repertoire of diacritical characters in CM fonts was insufficient for most languages using diacritical characters. The TEX accenting mechanism (the `\accent` primitive), meant as a solution to this problem, was unsatisfactory—for example, accents and hyphenation conflicted. The problem was recognized relatively early and various approaches were used to remedy the situation.

For example, the Polish extension of CM in the PK format was prepared in Poland (PL fonts, [8]), but this worked only for Polish TEXies.

Also worthy of note is the *European Computer Modern Fonts* (EC) project led by Jörg Knappen and Norbert Schwarz, triggered during the TEX Users Group conference in Cork, 1990, and finished in 1997 [16]. The EC metric files, however, are slightly incompatible with CM metric files, by circa 0.025%.[3]

A rather general technique was applied by Lars Engebretsen, who attempted to eliminate the necessity of using the `\accent` primitive by making virtual fonts, dubbed *Almost European* (AE), containing quite a large set of the European diacritical characters [1].

[3] The reason behind this discrepancy is a peculiarity of METAFONT arithmetic: the formula $\mathbf{1/36} * i$ yields different results than the formulas $i/\mathbf{36}$ and $\mathbf{1/36}i$; for example, for $i = \mathbf{3600}$ the results are $\mathbf{99.97559}$ for the first formula and $\mathbf{100}$ for the latter formulas. The first formula was used in the EC fonts (in the `gendef` macro).

Hyphenation worked with AE fonts, though still unsatisfactorily—for example, coinciding of such accents as cedilla or ogonek with a main glyph is inconvenient when a non-intersecting outline is required (for cutting plotters, for outlined titles, and so on). Moreover, the virtual fonts are obviously unusable outside the TEX realm.

Latin Modern fonts in the PostScript Type1 format

In 2002, during the EuroBachoTEX meeting, a proposal of converting Engebretsen's AE fonts into the PostScript Type 1 format and augmenting with them the set of necessary diacritical characters was put forward by representatives of European TEX user groups. We had no choice but to accept the proposal with delight.

Our initial plan was to use the AE fonts as our departure point; we even wanted to preserve the original name, *Almost European*, coined by Lars Engebretsen. It turned out, however, to be much more efficient to prepare the enhanced version of the CM fonts from scratch, and so sticking to Lars Engebretsen's name seemed inadequate, because the differences were too essential.

All in all, inspired by both EC and AE fonts, we came up with the *Latin Modern* (LM; see [11]) project which was accepted by the user groups.

Fortunately for us, freely available quality CM fonts in the PostScript Type1 format already existed. In the 1980s and 90s, they were produced (from traced bitmaps improved by very solicitous manual tuning) for commercial purposes by Blue Sky Research and Y&Y. Nearly a decade later, they were released to the public thanks to the efforts of the American Mathematical Society.[4]

We converted the PostScript Type 1 files of the CM fonts to MetaType 1 and wrote the METAPOST software relevant for generating the characters we decided to add (mainly diacritical letters). The work had already been partially done by Janusz M. Nowacki, who prepared the PostScript Type 1 version of the PL fonts in 1997. The official version of the LM fonts, 1.000, was eventually released in 2006 (in the meantime, several unofficial versions were released for testing purposes). The LM collection of fonts consisted of 72 text fonts, each counting about 700 glyphs, plus 20 CM-like math fonts.

[4] In 1997, a consortium of scientific publishers (American Mathematical Society, Elsevier Science, IBM Corporation, Society for Industrial and Applied Mathematics, and Springer-Verlag) in cooperation with Blue Sky Research and Y&Y decided to release these excellent fonts non-commercially; in order to assure the authenticity of the fonts, copyright was assigned to the American Mathematical Society (http://www.ams.org/publications/type1-fonts).

In 2009, an extensive revision of the LM fonts was carried out: the text fonts contain now more than 800 glyphs each (altogether more than 60,000 glyphs) and the glyphs conform to the changes introduced by Donald E. Knuth in 1992.

Almost all of the CM text fonts have counterparts in the LM family; the exceptions are one monospaced font, `cmtex10`, emulating Donald E. Knuth's keyboard layout, and the rarely used `cmff10`, `cmfi10`, `cmfib8`, and `cminch`. So far, nobody has complained about this inconsistency. Instead, encouraged by Hans Hagen, we decided to create 10 variants of typewriter LM fonts not having counterparts in the CM family: `lmtlc10`, `lmtk10`, `lmtl10`, `lmvtk10`, and `lmvtl10` (monospaced light condensed and monospaced and variable-width dark and light, respectively) plus their oblique variants `lmtko10`, `lmtlo10`, `lmtlco10`, `lmvtko10`, and `lmvtlo10`.

Latin Modern fonts in the OTF format

It was relatively easy to prepare the LM family of fonts in the OTF format using the AFDKO package: it mainly necessitated preparing a few extra data files in the *OpenType Feature File Specification* language [32]. Needless to say, the experience gathered at this stage came in handy during the work on the TG fonts (see page 24).

There was trouble, however, with the 20 math fonts. We provided the respective LM equivalents in PostScript Type 1 format. For compatibility with the (obsolete) PL fonts, the symbol fonts, `lmsy*` and `lmbsy*`, contain two extra glyphs: slanted greater-or-equal and less-or-equal signs, used traditionally in Polish math typography. As the math extension for OpenType did not exist yet, we decided not to convert these fonts to OTF. We knew that the companies that had invented and maintained the OTF standard, in cooperation with the American Mathematical Society and the Unicode Consortium, were working on extending the standard with math typesetting capabilities. We expected that by using the enhanced OTF specification we would be able to create a TEX-compatible math OTF collection. Alas, the Unicode Consortium report on Unicode support for mathematics [37], followed by the initially confidential Microsoft specification [29], snuffed out our hopes. It turned out that OTF math and TEX math cannot be reconciled. More information on the interrelationships between OTF and TEX math can be found in Ulrik Vieth's thorough analysis [24].

Repertoire issues

Our primary aim was to provide a repertoire of diacritical letters rich enough to cover all European languages. We thoroughly exploited Michael Everson's comprehensive study of European alphabets [2], as well as other sources. Several other languages using Latin-based alphabets, such as Vietnamese, Navajo and Pali, are covered.

Initially, contrary to the *Latin Modern* name, we considered including Cyrillic alphabets as well. Having thought the matter over, we decided, with regret, to abandon this idea and concentrated our efforts on OTF math fonts.

Besides diacritical characters, the Latin Modern fonts contain also a number of glyphs traditionally present in TEX fonts, such as Greek symbols, currency symbols, technical symbols, etc. A detailed description of the contents of the fonts can be found in the document entitled *Latin Modern Family of Fonts. The Technical Documentation*, included in the LM distribution package [11].

Two groups of glyphs are widely used in typography but neglected to a certain extent in the CM fonts, namely, *caps and small caps* and *old style numerals*, also known as *text figures* or *nautical digits*; the latter name originates from their widespread use in tables in nautical almanacs at one time. For reasons hard to explain, the caps and small caps were implemented in the CM family as a separate font, while the old style numerals (upright!) are in the math italic font (`cmmi*`).

The LM fonts incorporated caps and small caps from the CM family, together with its width idiosyncrasy: the `lmcsc10` font, like `cmcsc10`, has capital letters wider than the `lmr10` font by circa 8%. There are two caps and small caps fonts in the LM collection, namely, the regular and typewriter specimen, `lmcsc10` and `lmtcsc10`, as in CM, plus their oblique variants, `lmcsco10` and `lmtcsco10`, absent from CM. In principle, small caps glyphs could be transferred to other fonts, but we decided to not alter the framing of the original CM family, more so as CM has no sans-serif caps and small caps.

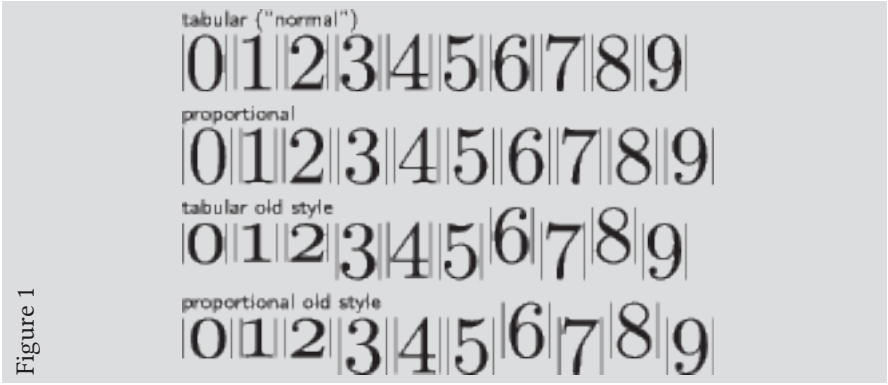


Figure 1

Concerning old style numerals, we could not accept the CM oddity and included them in all text fonts of the LM family. Further, all numerals come in 2 ›flavors‹: normal (fixed-width a.k.a. tabular) and proportional (variable-width, having balanced sidebearings) which altogether yields 4 variants—see Figure 1.

The TFM format contains only 256 slots for glyphs, thus, the whole repertoire of glyphs cannot be accessed at once if TEX is used in the a »traditional« way, that is, with TFM files. In particular, accessing the different kinds of numerals when using PostScript Type1 fonts plus TFM metric turns out to be clumsy; as a result, only tabular old style numerals are available in our package, in the TS1 encoding (see below). On the one hand, OTF fonts seem more convenient as they do not impose such a restriction; for example, all numerals can be accessed by using the OTF feature mechanism, more precisely, by the features onum, lnum, pnum, and tnum [33]. On the other hand, OTF metric data cannot, in general, be fully compatible with TFM metric data because the glyph widths in OTF fonts must be represented by integer quantities. This should be considered as drawback by TEXies—see page 22.

Following the LATEX tradition, we provided several encodings for the LM fonts, namely:

- CS (CS TUG) encoding (cs-*.tfm),
- EC (Cork) encoding (ec-*.tfm),
- L7X (Lithuanian) encoding (l7x-*.tfm),
- QX (GUST) encoding (qx-*.tfm),
- RM (»regular math«, used in OT1 and OT4) encodings (rm-*.tfm),
- Y&Y's TEX'n'ANSI a.k.a. LY1 encoding (texnansi-*.tfm),
- T5 (Vietnamese) encoding (t5-*.tfm),
- Text Companion for EC fonts a.k.a. TS1 (ts1-*.tfm).

The LATEX support for all these encodings, due to Marcin Woliński, is also part of the LM distribution.

TFM files nominally representing the same encoding do not always define the same set of characters; for example, the character sets of cmr10.tfm and cmtt10.tfm differ. The original CM fonts comprise 7 different character sets, with an idiosyncratic difference between the cmr10 and cmr5 layouts. As a remnant of the CM design, there are 5 different character sets of the LM text fonts:

1. 821 glyphs (basic set): lmb10 lmbo10 lmbx10 lmbx12 lmbx5 lmbx6 lmbx7 lmbx8 lmbx9 lmbxi10 lmbxo10 lmdunh10 lmduno10 lmr10 lmr12 lmr17 lmr5 lmr6 lmr7 lmr8 lmr9 lmri10 lmri12 lmri7 lmri8 lmri9 lmro10 lmro12 lmro17 lmro8 lmro9 lmss10 lmss12 lmss17 lmss8 lmss9 lmssbo10 lmssbx10 lmssdc10 lmssdo10 lmsso10 lmsso12 lmsso17 lmsso8 lmsso9 lmu10 lmvtk10 lmvtko10 lmvtl10 lmvtlo10 lmvtt10 lmvtto10
2. 824 glyphs: lmssq8 lmssqbo8 lmssqbx8 lmssqo8
   extra characters: varI varIJ varIogonek
3. 814 glyphs: lmcsc10 lmcsco10
   missing characters: f_k ff ffi ffl fi fl longs
4. 785 glyphs: lmtk10 lmtko10 lmtl10 lmtlc10 lmtlco10 lmtlo10 lmtt10 lmtt12

lmtt8 lmtt9 lmtti10 lmtto10
missing characters: f_k ff ffi ffl fi fl Germandbls hyphen.prop IJ ij permyriad servicemark suppress trademark varcopyright varregistered zero.oldstyle zero.prop one.oldstyle one.prop two.oldstyle two.prop three.oldstyle three.prop four.oldstyle four.prop five.oldstyle five.prop six.oldstyle six.prop seven.oldstyle seven.prop eight.oldstyle eight.prop nine.oldstyle nine.prop

5. 784 glyphs: lmtcsc10 lmtcso10
missing characters: as in 4, also longs

Compatibility issues

We did our best to provide outline fonts that can be used as a replacement for CM fonts. To a certain extent, we managed to achieve this goal, namely, the PostScript drivers which process TEX documents typeset with CM metric files, can use either CM or LM PostScript Type 1 fonts—special map files for PostScript Type 1 fonts are available for this purpose. The metric files, however, cannot be used replaceably, because the typesetting algorithms are intrinsically unstable—even tiny (rounding) errors may yield glaringly different results.

Therefore, LM users also cannot expect PostScript Type 1 and OTF fonts to be used replaceably. Recall that the OTF format requires integer number representation for glyph widths. The »reference« quantity is the *em* unit: 1 em = 2048 units for fonts using splines of the 2nd degree, 1 em = 1000 units for fonts using splines of the 3rd degree (e.g., our LM and TG fonts). Therefore, in our case, the difference in width is on average 1/2000 *em* (twice as large as the variation in the EC widths), that is, circa 0.005 *pt* for 10-point fonts.

Because the MetaType 1 sources of the LM fonts are the result of conversion from PostScript Type 1, the widths stored in the LM TFM files are not identical to the respective original CM widths. They are closer, however, to the original quantities by an order of magnitude as compared to the EC and OTF widths. At the cost of great effort (by referring to the METAFONT sources), we might have eliminated rounding errors in LM widths. But it would not cure the problem of (non-)replaceability, as widths are not the only source of trouble. Differences in heights and depths of glyphs may also yield unexpected behavior of the TEX typesetting algorithm.

The problem of heights and depths in TEX turns out to be unavoidable, and quite serious: the TFM format permits by design only 16 different heights and depths, including the obligatory entries containing the value zero. If there are in fact more heights and depths in a given font, their number is cleverly reduced to 16 by METAFONT (as well as by the METAPOST and TFtoPL programs). One of the

certainly unwanted results is that the same glyph in different encodings may have different heights and/or depths! For example, the height of the letter ›A‹ is 6.88875 pt in the `rm-lmr10.tfm` file (this layout is an extension to 256 slots of the `cmr10` layout), it is 6.99648 pt in the `t5-lmr10.tfm` file (Vietnamese layout), while in the canonical `cmr10.tfm` file it is 6.83331 pt.

This is not the end of the list of possible sources of incompatibility between CM and LM fonts. Positioning of the accents is also a long story. These and related aspects are explained in detail in [12].

Finally, let us consider a somewhat atypical example of incompatibility between the LM and CM fonts related to Donald E. Knuth's mistake in a CM `ligtable` program, uncorrectable for obvious reasons but basically harmless; namely, `roman.mf` contains the following:

```
% three degrees of kerning
k#:=-.5u#; kk#:=-1.5u#; kkk#:=-2u#;
ligtable "k":
  if serifs: "v": "a" kern -u#, fi
  "w": "e" kern k#, "a" kern k#,
       "o" kern k#, "c" kern k#;
```

The culprit is the `if serifs` clause: the kern pair ›ka‹ appears twice in the TFM files of serif fonts with the values $-u\#$ and $-0.5u\#$, respectively, as is easily seen in the following fragment of the `cmr10.pl` file (the respective lines are marked with arrows):

```
(CHARACTER C k
   (CHARWD R 0.527781)
   (CHARHT R 0.694445)
   (COMMENT
      (KRN C a R -0.055555) ←
      (KRN C e R -0.027779)
      (KRN C a R -.027779)  ←
      (KRN C o R -0.027779)
      (KRN C c R -0.027779)
      )
   )
```

Moreover, there are no ›va, ›vc‹, ›ve‹, and ›vo‹ kern pairs in sans-serif fonts, although there are ›kc‹, ›ka‹, ›ke‹, ›ko‹, ›wa‹, ›wc‹, ›we‹, and ›wo‹ kern pairs in these fonts. We could not see the reason for ignoring ›v‹ in this context, thus we decided to add the relevant kern pairs in the LM fonts; we also added quite a few other kern pairs missing, in our opinion, from the CM fonts, for example, ›eV‹ and ›kV‹.

Summing up, we believed that we had good reasons for giving up the struggle for a »100-percent compatibility« between LM and CM metrics, whatever that would mean, and to confine ourselves to providing the mentioned replaceability of outlines.

The TEX Gyre collection of fonts

Heartened by the results of the LM enterprise, we accepted without hesitation the next proposal: the »LMization« of the family of fonts provided by Ghostscript as a replacement for the renowned *Adobe base 35* fonts, generously released by the URW++ company under free software licenses.

- ITC Avant Garde Gothic (book, book oblique, demi, demi oblique)
- ITC Bookman (light, light italic, demi, demi italic)
- Courier (regular, regular oblique, bold, bold oblique)
- Helvetica (medium, medium oblique, bold, bold oblique)
- Helvetica Condensed (medium, medium oblique, bold, bold oblique)
- New Century Schoolbook (roman, roman italic, bold, bold italic)
- Palatino (regular, regular italic, bold, bold italic)
- Symbol
- Times (regular, regular italic, bold, bold italic)
- ITC Zapf Chancery (medium italic)
- ITC Zapf Dingbats

Since our aim was »LMization«, we excluded the Symbol and ITC Zapf Dingbats non-text fonts from the scope of our interest.

After a brief (but heated) debate, the name of the project and of its constituent fonts were coined. The project was dubbed TEX Gyre (TG) and the following names were accepted (the respective file name kernels, original Adobe names and Ghostscript, that is, URW, names are given in parentheses):

- TG Adventor (qag / ITC Avant Garde Gothic / URW Gothic L)
- TG Gyre Bonum (qbk / ITC Bookman / URW Bookman L)
- TG Cursor (qcr / Courier / Nimbus Mono L)
- TG Heros (qhv / Helvetica / Nimbus Sans L)
- TG Heros Condensed (qhvc / Helvetica Condensed / Nimbus Sans L Condensed)
- TG Schola (qcs / New Century Schoolbook / Century Schoolbook L)
- TG Pagella (qpl / Palatino / URW Palladio L)
- TG Termes (qtm / Times / Nimbus Roman No9 L)
- TG Chorus (qzc / ITC Zapf Chancery / URW Chancery L)

We initially considered including Cyrillic alphabets, but, as in the case of the LM fonts, we eventually abandoned this idea, also with regret, the more so as the

Ghostscript fonts at that time contained an (apparently unfinished) set of Cyrillic glyphs.

We expected that the main effort would be the making of extra glyphs plus maybe correcting outlines here and there. To our surprise, quite a few glyphs required tuning because of evident errors in outlines. One of the most striking examples is the glyph ›eight‹ from the URW Schoolbook bold font[5]—see Figure 2.

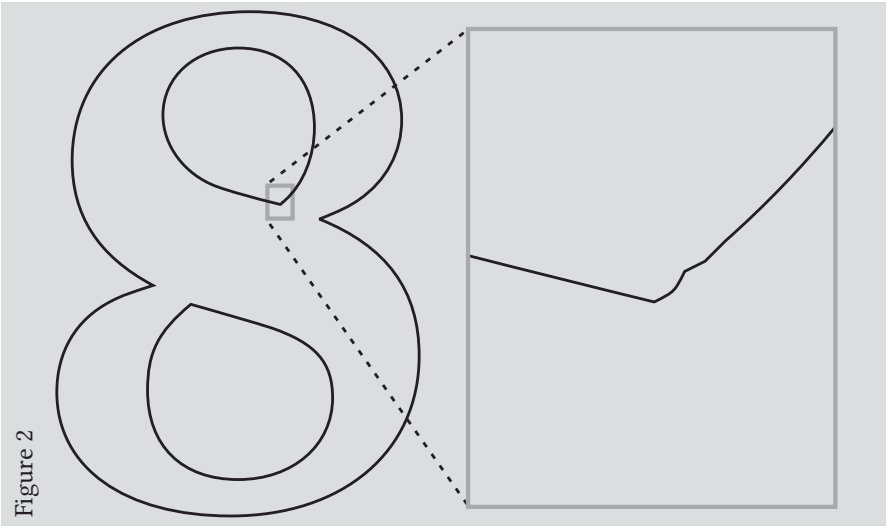


Figure 2

Mostly, we removed redundant or wrong nodes from the outline definitions, deleting more than 5% of them in all. In the case of TG Pagella, however, the insertion of extra nodes turned out to be necessary. The chart in Figure 3 shows some statistics for the upright TG fonts. The diagram concerns the version of the Ghostscript fonts which we used as our starting point. We used circa 350 glyphs from each font. The total number of nodes in these glyphs varied from circa 10,000 to 25,000 per font. In the current release, the TG text fonts count almost 1100 glyphs each with the number of nodes varying from circa 30,000 to 65,000 (for sans-serif and serif italic variants, respectively; see [14]).

[5] Recently, the font has been renamed to ›C059 bold'; the bug was removed from the Ghostscript distribution only in 2015, although the TEX Collection 2016 distribution still contains (due to the legacy reasons) the faulty glyph.

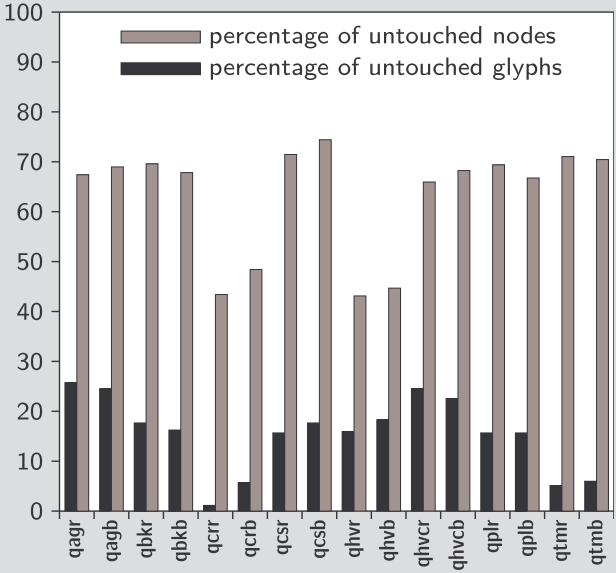


Figure 3

Repertoire issues

The difference in the number of glyphs between the TG and LM text fonts (the former having circa 250 glyphs more per font) is due mainly to the presence of small caps in the TG fonts (225 glyphs per font). Also unlike the LM fonts, each TG font contains the complete Greek alphabet and a few technical glyphs, such as ›lozenge‹ and ›lscript‹.[6] Except for these, the LM and TG fonts share the same repertoire of glyphs and the same set of TFM encodings (see page 19).

The LaTeX support for these encodings was also provided by Marcin Woliński.

Compatibility issues

The consistency of the widths of the original Adobe and the respective TG glyphs was one of our main concerns, as the TG fonts were meant as potential replacements for the Adobe fonts. It turned out, however, that the original font metric files [27], contained apparent metric flaws which we decided, with some hesitation, not to retain.

[6] For historical reasons, the LM fonts contain a few additional variants of the base, left, and right double quotes, absent from the TG fonts.

A typical example (concerning Helvetica, a.k.a. Nimbus Sans L, a.k.a. TG Heros) is depicted in Figure 4. Both Spanish ›¡‹ and Scandinavian ›ø‹ glyphs belong to the Adobe Standard Encoding set, hence one can expect that they should be considered important and thus unchangeable. Nevertheless, we could not see a reason for using widths different from the ›!' and ›o‹ widths, respectively, and certainly there is no substantiation for the asymmetry of sidebearings, especially in the case of ›ø'; therefore, we decided to alter the metrics.

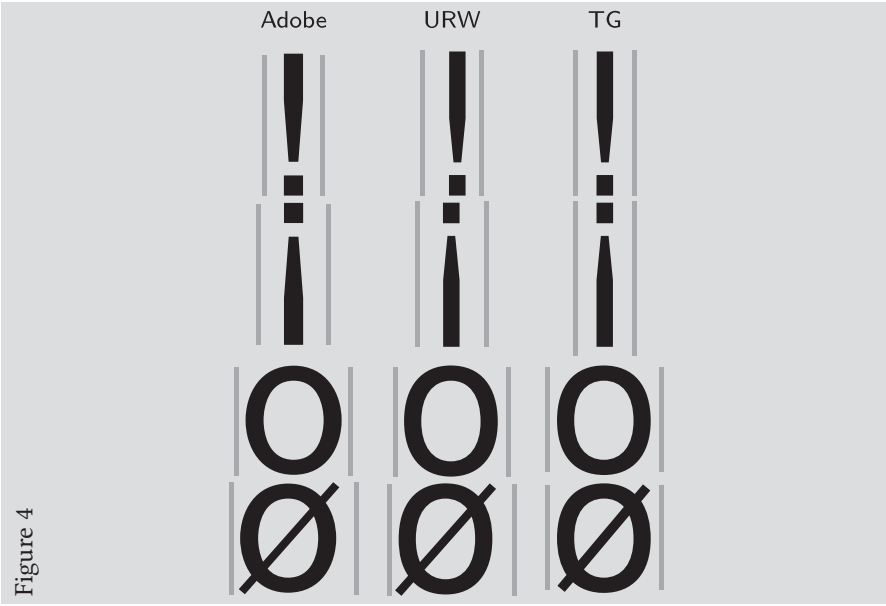


Figure 4

Fortunately, there are few such cases in the TG collection; each is mentioned in the documentation of the TG fonts [14].

Because the original widths for the TG fonts, unlike in the LM collection, were integer numbers, there is no metric discrepancy between the PostScript Type 1 and OTF font formats, as far as widths are concerned. Heights and depths, however, are subject to the same restrictions as discussed in page 22.

OTF math fonts

The chronic problem of lack of math support for the TG collection became the impetus for our third venture: math fonts for the LM and TG collections in the OTF format. The recent update of the LM and TG fonts took place at the end of 2009. The

math extension for the OTF format ([29]) had been released and there existed the FontForge font editor ([25]) capable of generating such fonts. So we embarked upon an expedition into unknown regions—since then we have focused our attention on the work on OTF math fonts, again with the benevolent encouragement and support from the TEX users groups.

As we should have expected, the task turned out interesting and absorbing, and, according to Hofstadter's Law,[7] we spent more time on it than we expected. From the very beginning, we aimed at making a collection of mutually consistent math OTF fonts and we underestimated the heterogeneity of the sources of additional alphabets and the problem of interrelationships—works on subsequent fonts entailed moving backwards to the fonts which we had prematurely considered ready. Nevertheless, in 2011, we happily announced the release of our first math OTF font, namely, Latin Modern Math. Altogether, six math fonts have been released by the GUST e-foundry so far [9, 10]:

- TG Latin Modern Math
- TG Bonum Math
- TG Schola Math
- TG Pagella Math
- TG Termes Math
- TG DejaVu Math

This amounts to nearly half of all OTF math fonts released in the world. Besides these, the following OTF math fonts have been released: Asana by Apostolos Syropoulos, Neo-Euler and XITS by Khaled Hosny, STIX by the STI Pub companies,[8] Cambria Math by Microsoft,[9] Lucida Math by Bigelow & Holmes, and Minion Math by Johannes Küster; the latter three fonts are distributed commercially.

OTF math font contents

Math OTF fonts, as we expounded in [10], are truly nasty beasts. In accordance with [35] and [37], they are expected to contain a plethora of glyphs: letters, arrows, math operators and delimiters, geometrical shapes, technical symbols, etc. The

[7] Hofstadter's Law: *It always takes longer than you expect, even when you take into account Hofstadter's Law*—Douglas R. Hofstadter.

[8] The STIX project began through the joint efforts of American Mathematical Society (AMS), American Institute of Physics Publishing (AIP), American Physical Society (APS), American Chemical Society (ACS), Institute of Electrical and Electronic Engineers (IEEE), and Elsevier Science; these companies are collectively known as the STI Pub companies.

[9] Cambria Math was the first math font published, conforming to the specification *MATH—The mathematical typesetting table* [29]. It was released by Microsoft in 2007, along with a MS Office version equipped with the capability of handling the math font and editing math formulas.

presence of some of them, particularly the (over)abundance of peculiar geometrical shapes and arrows, is hard to substantiate in our opinion.

Initially, we planned also releasing the math companion to the TEX Gyre sans-serif fonts, TG Adventor and TG Heros, but the Unicode specification for the contents of math fonts, [37], turned out definitely »serif-oriented«. Let us take, for example, the arrangement of the LM Math font shown in Table 1: following the cited specification, we combined several LM source fonts into a single complex font. As the table clearly shows, the basic subsets, that is, plain, bold, italic and bold italic are assumed to consist of serif glyphs by default. It is not obvious how the table should be adjusted to suit sans-serif math fonts. Work on this issue is in progress.

Table 1

category	charset	source fonts
plain (upright, serif)	L*, G, D	`lmr`, `lmmi` (upright)
bold	L, G, D	`lmbx`, `lmmib` (upright)
italic	L, G	`lmmi`
bold italic	L, G	`lmmib`
sans-serif	L, D	`lmss`
sans-serif bold	L, G, D	`lmssbx`
sans-serif italic	L	`lmsso`
sans-serif bold italic	L, G	`lmssbo`
calligraphic	L	`eusm` (slanted)
bold calligraphic	L	`eusb` (slanted)
Fraktur	L	`eufm`
bold Fraktur	L	`eufb`
double-struck	L, D	`bbold` (by Alan Jeffrey)
monospace	L, D	`lmtt`

L, G, D—Latin, Greek and digits, respectively
L*—contains also diacritical letters and punctuation
All the alphanumeric glyphs specified in the table, except for the »plain« ones (first row), are given special mathematical Unicode slots—see [37]

The original CM fonts, and thus the LM fonts, do not contain the complete sans-serif Greek. We generated the missing glyphs using modified METAFONT sources in order to generate outlines instead of bitmaps and tuning the result manually, if required.

Moreover, a math font is bound to contain plenty of other characters, most notably glyphs used for subscripts of the 1st and 2nd order, used also for superscripts; we'll refer to them shortly *pars pro toto* as subscripts. They are accessed by the OTF feature mechanism, more precisely by the math extension feature `ssty` [33].

Extensible symbols, like large brackets or radicals, are another important group of math-oriented glyphs. An extensible symbol consists of a collection of a few

components (the left part of Figure 5) assembled by the typesetting engine into a seemingly single character (the right part of Figure 5).

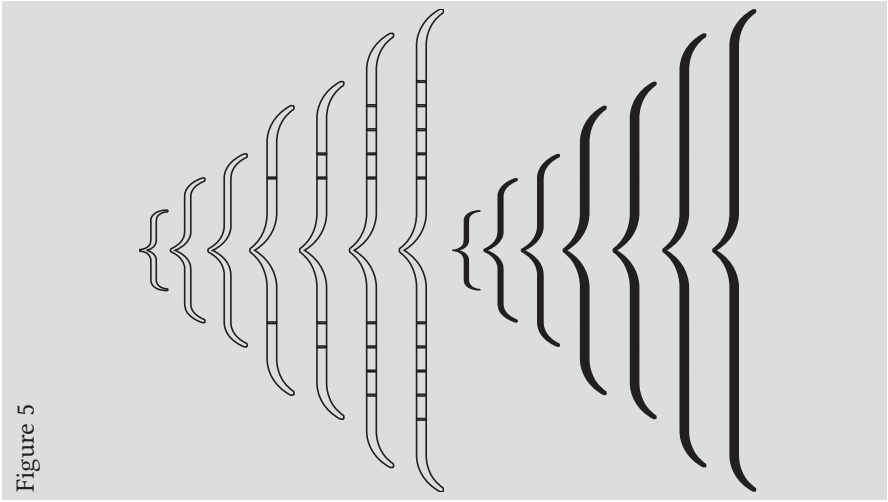


Figure 5

First, a glyph of adequate size is searched for in the so-called chain of glyph variants (here: the three leftmost curly braces). If a proper glyph is not found, the typesetting engine assembles a respectively large symbol from the relevant pieces using a fairly complex algorithm—everybody who has attempted unsuccessfully to fit braces around a formula according to one's desire probably knows it.

In the case of the radical symbol, the situation is still more complex, because the OTF and TEX geometric structure, as well as the relevant metric data of the components differ, as is shown in Figure 6 which visualises »stages« of the assembling of an extensible radical symbol.

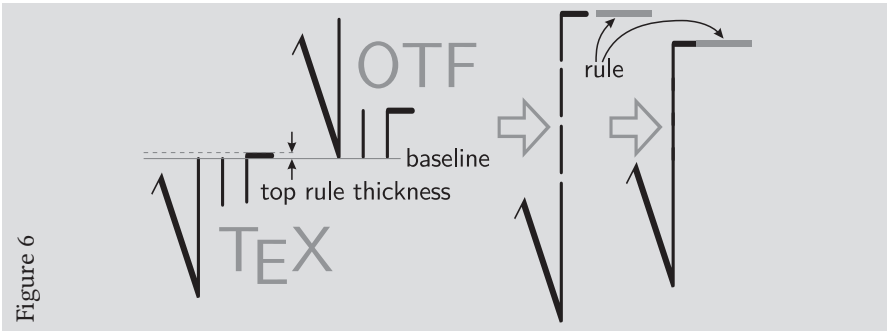


Figure 6

In both cases, the radical symbol is assembled from glyphs taken from a relevant font and a line (rule) drawn by the typesetting program at the top of the symbol (marked with a gray color). The thickness of the rule, however, is given explicitly as a parameter in math OTF files, while it is inferred from the height of the top element of the radical symbol in TEX (recall the problem of the limited number of heights in a TEX metric file).

There is an important difference between the TEX and OTF math font specification concerning extensible glyphs: TEX is equipped with the ability to assemble compound glyphs from pieces only vertically, while the OTF format offers both horizontal and vertical assembling. In TEX (i.e., in the `plain` TEX format), such glyphs as horizontal braces are defined by special macros using rules and a few glyphs from a relevant font:

```
\def\downbracefill
  {$\m@th \setbox\z@\hbox{$\braceld$}%
  \braceld\leaders\vrule height\ht\z@
     depth\z@\hfill\braceru
  \bracelu\leaders\vrule height\ht\z@
     depth\z@\hfill\bracerd$}
\def\upbracefill [...]
```

Incidentally, it seems that diagonal extensible glyphs have not yet been invented. Could it be that the conundrum is too difficult to solve?

More on the differences between the TEX and OTF specifications of the structure of math fonts can be found in Ulrik Vieth's survey [24].

In the case of LM Math, we were fortunate to have well-known clean sources for a base, already containing 7-point and 5-point variants, suitable for typesetting subscripts, and components for assembling extensible glyphs.

We have to confess, however, that we were not especially delighted with the Computer Modern calligraphic script. More pleasingly designed, to our eyes, are the calligraphic letters of the renowned Euler family. Therefore, we decided to transfer the glyphs from the Euler fonts (slanting them slightly) to the LM Math font:

In the case of the TG math fonts, the situation was slightly worse. The basic sources, that is, the text fonts, were already improved by us, but the sources of the relevant additional character sets were highly heterogeneous. We used freely available fonts

of the best possible quality as our base; nevertheless, much manual tuning was necessary (recall the tuning of the sources of the TG text fonts).

Moreover, not all suitable fonts had acceptable free licenses. In a few cases, we had to contact the authors personally. It should be emphasized that in all cases the authors, if we managed to reach them, courteously agreed to make their fonts available for our purposes. The following external fonts were used in the TG math fonts project (in alphabetical order):

- Lato by Łukasz Dziedzic (TG Schola Math)
- Kerkis by Apostolos Syropoulos and Antonis Tsolomitis (TG Bonum Math)
- Leipziger Fraktur replica by Peter Wiegel (TG Bonum Math and TG Termes Math)
- Math Pazo by Diego Puga (TG Pagella Math)
- Odstemplik by Grzegorz Luk (gluk) (TG Pagella Math)
- Theano Modern by Alexey Kryukov (TG Schola Math)

We are most grateful to all the authors for their prominent aid.

Visual issues

Observe that the same monospaced alphanumeric characters (excerpted from TG Cursor) are shared by TG Bonum Math, TG Schola Math, and TG Termes Math. In such cases one must carefully check whether the size of the glyphs being included fits the size of the basic set of glyphs. Nominally, all fonts (both in the PostScript Type1 and OTF formats) have the same design size, 10 typographic points (recall that 1 point=1/72 inch; cf. page 22). Working on the TG math fonts, we had to adjust the size of the subsets a few times. For example, we enlarged the borrowed monospaced glyphs to circa 112.5% in TG Schola Math; otherwise the monospaced alphabet looked too small in combination with Schola's brawny glyphs.

The visual harmonizing of the supplementary alphabets with the main font face concerns not only alphanumeric glyphs. Even essentially geometrical shapes should also reflect the characteristic features of the main font, for example, the thickness of stems, the ending of arms, etc. Seemingly trivial glyphs, such as arrows, serve as a convenient example: they have slightly different shapes in each of our math fonts—see Figure 7.

Another example is the shape of integrals. Historically, the integral symbol originates from the letter ›long s'‹[10] which is nowadays identical with a barless ›f‹.

[10] The cursive long ›s‹ for denoting an integral operation, i.e., infinite summation, was introduced by Gottfried Wilhelm Leibniz in 1675 in an unpublished paper *Analyseos tetragonisticæ pars secunda* (*Second part of analytical quadrature*).

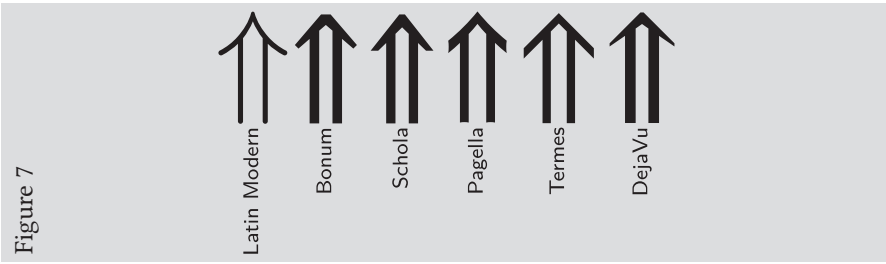


Figure 7

Therefore, we did our best to preserve some characteristic features of the letter ›f‹ in the design of the integral shape—see Figure 8.

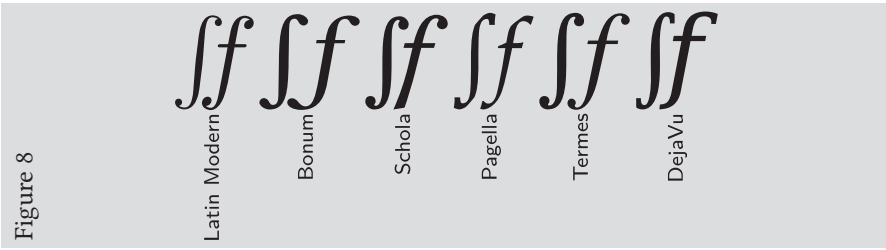


Figure 8

We are not going to dwell on the visual aspects of the math font design, as the number of details relevant for a font with over 4000 glyphs (and counting) would perhaps become rather overwhelming. Notwithstanding, one aspect needs emphasizing, namely, the problem of sidebearings and kerning for alphanumeric symbols.

It is generally accepted by typographers that math symbols should have larger sidebearings than the respective text glyphs. In particular, TEX math italic glyphs are a little bit broader and have larger sidebearings than the text italic glyphs. It is not obvious, however, how large such sidebearings should be. We have already experimented with a few sizes but further tuning will probably be necessary. Of course, the broadening of sidebearings should not be applied to the basic font, that is, regular upright, because of multiletter names of functions and operators, such as ›sin‹ or ›max‹, which are traditionally typeset with regular upright letters.

As regards the problem of kerning, we decided not to include kerns in math fonts, although providing kerns for the upright regular alphabet might be reasonable. Actually, we consider introducing special math kerning (so-called »staircase« kerns a.k.a. »cut-ins«; see page 42). Thankfully, nobody has complained yet because of the lack of kerning in our math OTF fonts.

Repertoire issues

At present, a common standard for the repertoire of characters that should be present in an OTF math font does not exist. After several debates (mostly during TEX conferences) and many experiments, we came up with the tentative repertoire scheme presented in Table 2, which can be considered a detailed specification of Table 1.

Table 2

B – basic letters, A – accented letters, G – Greek letters,
D – digits, O – other symbols, P – punctuation

	B	A	G	D	O	P
plain (upright, serif)	+s	+s	+ds	+s	+	+s
italic	+s		+s			
bold	+s		+ds	+s		
bold italic	+s		+s			
sans-serif	+			+		
sans-serif italic	+					
sans-serif bold	+		+	+		
sans-serif bold italic	+		+			
calligraphic	+					
bold calligraphic	+					
Fraktur	+					
bold Fraktur	+					
double-struck	+			+		
monospace	+			+		

d—digamma excluded from relevant Unicode blocks
s—subscripts are to be added

We would like all TG math fonts (Bonum, DejaVu, Pagella, Schola, and Termes) to share this pattern. For LM Math, however, we adopted another scheme because of legacy concerns. Currently, LM Math contains circa 4800 glyphs while the remaining fonts contain circa 4250 glyphs each. LM Math contains more subscripts, as the original sources already provided them. On the other hand, the TG math fonts contain more size variants of integral symbols. This yields circa 550 glyphs more (net) in LM Math. At the moment, TEX Gyre DejaVu Math contains (in accordance with Table 2) subscript variants for bold upright glyphs, while the remaining TG math fonts need to be complemented with these glyphs. It is a typical instance of the frequently occurring »backtracking« procedures: the final decision was undertaken only after a few fonts had been released.

Concerning subscripts, it should be emphasized that subscript glyphs for the TG math fonts are obtained using the approach applied in the METAFONT sources of the Euler family of fonts, that is, by non-uniform rescaling of the respective normal-size glyphs. In a way, such »optical scaling« can be considered undesirable;

nevertheless, it proved acceptable for the Euler fonts, thus we decided to apply it also to the TG math fonts.

Some rather quirky characters which received Unicode slots are spaces. Unicode [35] defines quite a few space-related glyphs:

```
 0008 BACKSPACE (<control>)
*0020 SPACE
*00A0 NO-BREAK SPACE
*2002 EN SPACE
*2003 EM SPACE
 1361 ETHIOPIC WORDSPACE
 1680 OGHAM SPACE MARK
*2004 THREE-PER-EM SPACE
*2005 FOUR-PER-EM SPACE
*2006 SIX-PER-EM SPACE
*2007 FIGURE SPACE
*2008 PUNCTUATION SPACE
*2009 THIN SPACE
*200A HAIR SPACE
*200B ZERO WIDTH SPACE
*202F NARROW NO-BREAK SPACE
*205F MEDIUM MATHEMATICAL SPACE
 2408 SYMBOL FOR BACKSPACE
 2420 SYMBOL FOR SPACE
 3000 IDEOGRAPHIC SPACE
 303F IDEOGRAPHIC HALF FILL SPACE
*FEFF ZERO WIDTH NO-BREAK SPACE
 1DA7F SIGNWRITING LOCATION-WALLPLANE SPACE 1DA80 SIGNWRITING LOCATION-FLOORPLANE SPACE
 E0020 TAG SPACE
```

We decided to include a subset of the Unicode-defined spaces (marked with asterisks in the above list), even though their meaning and usage seems vague. The problem of spaces touches a general problem of the relation between the Unicode standard and typography. We discuss this topic in more detail in the next section.

The next two pages show the representative subset (for LM Math and TG Termes Math) of the repertoire we adopted as the GUST e-foundry »private standard«; gray squares denote zero-width characters. As one can see, the repertoires are very similar. One can assume that the difference in the repertoire will be imperceptible in practical applications.

Glyph repertoire of Latin Modern Math

ASCII: ! " # $ % & ' ( ) * + , - . / 0 1 2 3 4 5 6 7 8 9 : ; < = > ? @ A B C D E F G H I J K L M N O P Q R S T U V W X Y Z [\ ] ^ _ ` a b c d e f g h i j k l m n o p q r s t u v w x y z { | } ~

Latin-1: ¡ ¢ £ ¤ ¥ ¦ § ¨ © ª « ¬ ® ¯ ° ± ´ µ ¶ · ¸ º » ¿ À Á Â Ã Ä Å Æ Ç È É Ê Ë Ì Í Î Ï Ð Ñ Ò Ó Ô Õ Ö × Ø Ù Ú Û Ü Ý Þ ß à á â ã ä å æ ç è é ê ë ì í î ï ð ñ ò ó ô õ ö ÷ ø ù ú û ü ý þ ÿ

Latin-1 Supplement: Ā ā Ă ă Ą ą Ć ć Ĉ ĉ Ď ď Đ đ Ē ē Ė ė Ę ę Ě ě Ğ ğ Ġ ġ Ī ī Į į İ ı IJ ij Ķ ķ Ĺ ĺ Ļ ļ Ľ ľ Ł ł Ń ń Ņ ņ Ň ň Ŋ ŋ Ō ō Ő ő Œ œ Ŕ ŕ Ŗ ŗ Ř ř Ś ś Ş ş Š š Ţ ţ Ť ť Ũ ũ Ū ū Ŭ ŭ Ů ů Ű ű Ų ų Ÿ Ź ź Ż ż Ž ž ſ Ơ ơ Ư ư Ș ș Ț ț ȷ ˆ ˇ ˘ ˙ ˚ ˛ ˜ ˝ ‾

Combining Diacritical Marks:

Greek: Α Β Γ Δ Ε Ζ Η Θ Ι Κ Λ Μ Ν Ξ Ο Π Ρ Σ Τ Υ Φ Χ Ψ Ω α β γ δ ε ζ η θ ι κ λ μ ν ξ ο π ρ ς σ τ υ φ χ ψ ω ϑ ϕ ϖ ϰ ϱ ϴ ϵ

Latin Extended Additional: Ạ ạ Ả ả Ấ ấ Ầ ầ Ẩ ẩ Ẫ ẫ Ậ ậ Ắ ắ Ằ ằ Ẳ ẳ Ẵ ẵ Ặ ặ Ẹ ẹ Ẻ ẻ Ẽ ẽ Ế ế Ề ề Ể ể Ễ ễ Ệ ệ Ỉ ỉ Ị ị Ọ ọ Ỏ ỏ Ố ố Ồ ồ Ổ ổ Ỗ ỗ Ộ ộ Ớ ớ Ờ ờ Ở ở Ỡ ỡ Ợ ợ Ụ ụ Ủ ủ Ứ ứ Ừ ừ Ử ử Ữ ữ Ự ự Ỳ ỳ Ỵ ỵ Ỷ ỷ Ỹ ỹ

General Punctuation:

Currency Symbols: ₡ €

Combining Diacritical Marks for Symbols:

Letterlike Symbols: ℂ ℃ ℇ ℉ ℋ ℌ ℍ ℎ ℏ ℐ ℑ ℒ ℓ ℕ № ℗ ℘ ℙ ℚ ℛ ℜ ℝ ℠ ™ ℤ Ω ℧ ℨ K Å ℬ ℭ ℮ ℰ ℱ ℳ ℵ ℶ ℷ ℸ ℻ ℾ ℿ ⅀ ⅅ ⅆ ⅇ ⅈ ⅉ

Arrows: ← ↑ → ↓ ↔ ↕ ↖ ↗ ↘ ↙ ↚ ↛ ↞ ↠ ↢ ↣ ↦ ↤ ↥ ↧ ↩ ↪ ↫ ↬ ↭ ↮ ↰ ↱ ↲ ↳ ↶ ↷ ↺ ↻ ↼ ↽ ↾ ↿ ⇀ ⇁ ⇂ ⇃ ⇄ ⇅ ⇆ ⇇ ⇈ ⇉ ⇊ ⇋ ⇌ ⇍ ⇎ ⇏ ⇐ ⇑ ⇒ ⇓ ⇔ ⇕ ⇖ ⇗ ⇘ ⇙ ⇚ ⇛ ⇜ ⇝ ⇞ ⇟ ⇠ ⇡ ⇢ ⇣ ⇦ ⇧ ⇨ ⇩ ⇪ ⇫ ⇬ ⇭ ⇮ ⇯ ⇰ ⇱ ⇲ ⇳

Mathematical Operators: ∀ ∁ ∂ ∃ ∄ ∅ ∆ ∇ ∈ ∉ ∊ ∋ ∌ ∍ ∎ ∏ ∐ ∑ − ∓ ∔ ∕ ∖ ∗ ∘ ∙ √ ∝ ∞ ∟ ∠ ∢ ∣ ∤ ∥ ∦ ∧ ∨ ∩ ∪ ∫ ∬ ∭ ∯ ∰ ∱ ∲ ∳ ∴ ∵ ∶ ∷ ∸ ∹ ∺ ∻ ∼ ∽ ∾ ∿ ≀ ≁ ≂ ≃ ≄ ≅ ≆ ≇ ≈ ≉ ≊ ≋ ≌ ≍ ≎ ≏ ≐ ≑ ≒ ≓ ≔ ≕ ≖ ≗ ≘ ≙ ≚ ≛ ≜ ≝ ≞ ≟ ≠ ≡ ≢ ≣ ≤ ≥ ≦ ≧ ≨ ≩ ≪ ≫ ≬ ≭ ≮ ≯ ≰ ≱ ≲ ≳ ≴ ≵ ≶ ≷ ≸ ≹ ≺ ≻ ≼ ≽ ≾ ≿ ⊀ ⊁ ⊂ ⊃ ⊄ ⊅ ⊆ ⊇ ⊈ ⊉ ⊊ ⊋ ⊌ ⊍ ⊎ ⊏ ⊐ ⊑ ⊒ ⊓ ⊔ ⊕ ⊖ ⊗ ⊘ ⊙ ⊚ ⊛ ⊜ ⊝ ⊞ ⊟ ⊠ ⊡ ⊢ ⊣ ⊤ ⊥ ⊦ ⊧ ⊨ ⊩ ⊪ ⊫ ⊬ ⊭ ⊮ ⊯ ⊰ ⊱ ⊲ ⊳ ⊴ ⊵ ⊶ ⊷ ⊸ ⊹ ⊺ ⊻ ⊼ ⊽ ⊾ ⊿ ⋀ ⋁ ⋂ ⋃ ⋄ ⋆ ⋇ ⋈ ⋉ ⋊ ⋋ ⋌ ⋎ ⋏ ⋐ ⋑ ⋒ ⋓ ⋔ ⋕ ⋖ ⋗ ⋘ ⋙ ⋚ ⋛ ⋜ ⋝ ⋞ ⋟ ⋠ ⋡ ⋢ ⋣ ⋤ ⋥ ⋦ ⋧ ⋨ ⋩ ⋪ ⋫ ⋬ ⋭ ⋮ ⋯ ⋰ ⋱

Miscellaneous Technical: ⌀ ⌅ ⌆ ⌈ ⌉ ⌊ ⌋ ⌐ ⌙ ⌜ ⌝ ⌞ ⌟ ⌠ ⌡ ⌢ ⌣ ⟨ ⟩ ⌦ ⌧ ⌫ ⌬ ⌭ ⌮ ⌯ ⌰ ⌱ ⌲ ⌳ ⌴ ⌵ ⌶ ⌷ ⌸ ⌹ ⌺ ⌻ ⌼ ⌽ ⌾ ⌿ ⍀ ⍁ ⍂ ⍃ ⍄ ⍅ ⍆ ⍇ ⍈ ⍉ ⍊ ⍋ ⍌ ⍍ ⍎ ⍏ ⍐ ⍑ ⍒ ⍓ ⍔ ⍕ ⍖ ⍗ ⍘ ⍙ ⍚ ⍛ ⍜ ⍝ ⍞ ⍟ ⍠ ⍡ ⍢ ⍣ ⍤ ⍥ ⍦ ⍧ ⍨ ⍩ ⍪ ⍫ ⍬ ⍭ ⍮ ⍯ ⍰ ⍱ ⍲ ⍳ ⍴ ⍵ ⍶ ⍷ ⍸ ⍹ ⍺

Box Drawing, Block Elements: ─ │ ┌ ┐ └ ┘ ┬ ┴ ┼ ▁ ■ ▒ ▓

Geometric Shapes: ■ □ ▪ ▫ ▬ ▭ ▲ △ ▶ ▷ ▼ ▽ ◀ ◁ ◊ ○ ● ◦ ◯

Miscellaneous Symbols, Dingbats: ♠ ♡ ♢ ♣ ♤ ♥ ♦ ♧ ♩ ♭ ♮ ♯ ✓ ✠ ❙ ➡

Miscellaneous Mathematical Symbols-A: ⟂ ⟘ ⟙ ⟛ ⟜ ⟝ ⟞ ⟟ ⟡ ⟢ ⟣ ⟤ ⟥ ⟦ ⟧ ⟨ ⟩ ⟪ ⟫ ⟮ ⟯

Supplemental Arrows-A, B: ⟰ ⟱ ⟲ ⟳ ⟴ ⟵ ⟶ ⟷ ⟸ ⟹ ⟺ ⟻ ⟼ ⟽ ⟾ ⟿ ⤀ ⤁ ⤂ ⤃ ⤄ ⤅ ⤆ ⤇ ⤈ ⤉ ⤊ ⤋ ⤌ ⤍ ⤎ ⤏ ⤐ ⤑ ⤒ ⤓ ⤔ ⤕ ⤖ ⤗ ⤘ ⤙ ⤚ ⤛ ⤜ ⤝ ⤞ ⤟ ⤠ ⤡ ⤢ ⤣ ⤤ ⤥ ⤦ ⤧ ⤨ ⤩ ⤪ ⤫ ⤬ ⤭ ⤮ ⤯ ⤰ ⤱ ⤲ ⤳ ⤴ ⤵ ⤶ ⤷ ⤸ ⤹ ⤺ ⤻ ⤼ ⤽ ⤾ ⤿ ⥀ ⥁ ⥂ ⥃

Supplemental Mathematical Operators: ⨀ ⨁ ⨂ ⨃ ⨄ ⨅ ⨆ ⨇ ⨈ ⨉ ⨊ ⨋ ⨌ ⨍ ⨎ ⨏ ⨐ ⨑ ⨒ ⨓ ⨔ ⨕ ⨖ ⨗ ⨘ ⨙ ⨚ ⨛ ⨜ ⨝ ⨞ ⨟ ⨠ ⨡ ⨢ ⨣ ⨤ ⨥ ⨦ ⨧ ⨨ ⨩ ⨪ ⨫ ⨬ ⨭ ⨮ ⨯ ⨰ ⨱ ⨲ ⨳ ⨴ ⨵ ⨶ ⨷ ⨸ ⨹ ⨺ ⨻ ⨼ ⨽ ⨾ ⨿ ⩀ ⩁ ⩂ ⩃ ⩄ ⩅ ⩆ ⩇ ⩈ ⩉ ⩊ ⩋ ⩌ ⩍ ⩎ ⩏ ⩐ ⩑ ⩒ ⩓ ⩔ ⩕ ⩖ ⩗ ⩘ ⩙ ⩚ ⩛ ⩜ ⩝ ⩞ ⩟ ⩠ ⩡ ⩢ ⩣ ⩤ ⩥ ⩦ ⩧ ⩨ ⩩ ⩪ ⩫ ⩬ ⩭ ⩮ ⩯ ⩰ ⩱ ⩲ ⩳ ⩴ ⩵ ⩶ ⩷ ⩸ ⩹ ⩺ ⩻ ⩼ ⩽ ⩾ ⩿ ⪀ ⪁ ⪂ ⪃ ⪄ ⪅ ⪆ ⪇ ⪈ ⪉ ⪊ ⪋ ⪌ ⪍ ⪎ ⪏ ⪐ ⪑ ⪒ ⪓ ⪔ ⪕ ⪖ ⪗ ⪘ ⪙ ⪚ ⪛ ⪜ ⪝ ⪞ ⪟ ⪠ ⪡ ⪢ ⪣ ⪤ ⪥ ⪦ ⪧ ⪨ ⪩ ⪪ ⪫ ⪬ ⪭ ⪮ ⪯ ⪰ ⪱ ⪲ ⪳ ⪴ ⪵ ⪶ ⪷ ⪸ ⪹ ⪺ ⪻ ⪼ ⪽ ⪾ ⪿

Miscellaneous Symbols and Arrows: ⭀ ⭁ ⭂ ⭃ ⭄ ⭅ ⭆ ⭇ ⭈ ⭉ ⭊ ⭋ ⭌

CJK Symbols and Punctuation: 〖 〗

Alphabetic Presentation Forms: ff fi fl ffi ffl

Mathematical Alphanumeric Symbols: **A B C D E F G H I J K L M N O P Q R S T U V W X Y Z a b c d e f g h i j k l m n o p q r s t u v w x y z** *A B C D E F G H I J K L M N O P Q R S T U V W X Y Z a b c d e f g i j k l m n o p q r s t u v w x y z* ***A B C D E F G H I J K L M N O P Q R S T U V W X Y Z a b c d e f g h i j k l m n o p q r s t u v w x y z*** *A C D G J K N O P Q S T U V W X Y Z* ***A B C D E F G H I J K L M N O P Q R S T U V W X Y Z a b c d e f g h i j k l m n o p q r s t u v w x y z*** A B D E F G I J K L M O S T U V W X Y a b c d e f g h i j k l m n o p q r s t u v w x y z A B D E F G I J K L M O S T U V W X Y a b c d e f g h i j k l m n o p q r s t u v w x y z A B C D E F G H I J K L M N O P Q R S T U V W X Y Z a b c d e f g h i j k l m n o p q r s t u v w x y z *A B C D E F G H I J K L M N O P Q R S T U V W X Y Z a b c d e f g h i j k l m n o p q r s t u v w x y z* ***A B C D E F G H I J K L M N O P Q R S T U V W X Y Z a b c d e f g h i j k l m n o p q r s t u v w x y z*** A B C D E F G H I J K L M N O P Q R S T U V W X Y Z a b c d e f g h i j k l m n o p q r s t u v w x y z ı ȷ **A B Γ Δ E Z H Θ I K Λ M N Ξ O Π P Θ Σ T Υ Φ X Ψ Ω ∇ α** β γ δ ε ζ η θ ι κ λ μ ν ξ ο π ρ ς σ τ υ φ χ ψ ω ∂ ϵ ϑ ϰ ϕ ϱ ϖ *A B Γ Δ E Z H Θ I K Λ M N Ξ O Π P Θ Σ T Υ Φ X Ψ Ω ∇ α β γ δ ε ζ η θ ι κ λ μ ν ξ ο π ρ ς σ τ υ φ χ ψ ω ∂ ϵ ϑ ϰ ϕ ϱ ϖ* ***A B Γ Δ E Z H Θ I K Λ M N Ξ O Π P Θ Σ T Υ Φ X Ψ Ω ∇ α β γ δ ε ζ η θ ι κ λ μ ν ξ ο π ρ ς σ τ υ φ χ ψ ω ∂ ϵ ϑ ϰ ϕ ϱ ϖ*** **A B Γ Δ E Z H Θ I K Λ M N Ξ O Π P Θ Σ T Υ Φ X Ψ Ω ∇ α β γ δ ε ζ η θ ι κ λ μ ν ξ ο π ρ ς σ τ υ φ χ ψ ω ∂ ϵ ϑ ϰ ϕ ϱ ϖ** ***A B Γ Δ E Z H Θ I K Λ M N Ξ O Π P Θ Σ T Υ Φ X Ψ Ω ∇ α β γ δ ε ζ η θ ι κ λ μ ν ξ ο π ρ ς σ τ υ φ χ ψ ω ∂ ϵ ϑ ϰ ϕ ϱ ϖ*** **0 1 2 3 4 5 6 7 8 9** 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 **0 1 2 3 4 5 6 7 8 9** 0 1 2 3 4 5 6 7 8 9

Glyph repertoire of TEX Gyre Termes Math

ASCII: ! " # $ % & ' ( ) * + , - . / 0 1 2 3 4 5 6 7 8 9 : ; < = > ? @ A B C D E F G H I J K L M N O P Q R S T U V W X Y Z [\ ] ^ _ ` a b c d e f g h i j k l m n o p q r s t u v w x y z { | } ~

Latin-1: ¡ ¢ £ ¤ ¥ ¦ § ¨ © ª « ¬ ® ¯ ° ± ´ µ ¶ · ¸ º » ¿ À Á Â Ã Ä Å Æ Ç È É Ê Ë Ì Í Î Ï Ð Ñ Ò Ó Ô Õ Ö × Ø Ù Ú Û Ü Ý Þ ß à á â ã ä å æ ç è é ê ë ì í î ï ð ñ ò ó ô õ ö ÷ ø ù ú û ü ý þ ÿ

Latin-1 Supplement: Ā ā Ă ă Ą ą Ć ć Č č Ď ď Đ đ Ē ē Ė ė Ę ę Ě ě Ğ ğ Ģ ģ Ī ī Ī ī Į į İ ı Ĳ ĳ Ķ ķ Ĺ ĺ Ļ ļ Ľ ľ Ł ł Ń ń Ņ ņ Ň ň Ŋ ŋ Ō ō Ő ő Œ œ Ŕ ŕ Ŗ ŗ Ř ř Ś ś Ş ş Š š Ţ ţ Ť ť Ũ ũ Ū ū Ů ů Ű ű Ų ų Ÿ Ź ź Ż ż Ž ž ſ ƒ Ơ ơ Ư ư Ș ș Ț ț ȷ ˆ ˇ ˘ ˙ ˚ ˛ ˜ ˝

Combining Diacritical Marks: ̀ ́ ̂ ̃ ̄ ̅ ̆ ̇ ̈ ̉ ̊ ̋ ̌ ̏ ̑ ̣ ̦ ̧ ̨ ̬ ̭ ̮ ̯ ̰ ̱ ̲ ̳ ̸ ̿ ͍

Greek: Α Β Γ Δ Ε Ζ Η Θ Ι Κ Λ Μ Ν Ξ Ο Π Ρ Σ Τ Υ Φ Χ Ψ Ω α β γ δ ε ζ η θ ι κ λ μ ν ξ ο π ρ ς σ τ υ φ χ ψ ω ϑ ϕ ϖ ϰ ϱ ϴ ϵ

Latin Extended Additional: Ạ ạ Ả ả Ấ ấ Ầ ầ Ẩ ẩ Ẫ ẫ Ậ ậ Ắ ắ Ằ ằ Ẳ ẳ Ẵ ẵ Ặ ặ Ẹ ẹ Ẻ ẻ Ẽ ẽ Ế ế Ề ề Ể ể Ễ ễ Ệ ệ Ỉ ỉ Ị ị Ọ ọ Ỏ ỏ Ố ố Ồ ồ Ổ ổ Ỗ ỗ Ộ ộ Ớ ớ Ờ ờ Ở ở Ỡ ỡ Ợ ợ Ụ ụ Ủ ủ Ứ ứ Ừ ừ Ử ử Ữ ữ Ự ự Ỳ ỳ Ỵ ỵ Ỷ ỷ Ỹ ỹ

General Punctuation: ‐ ‒ – — ― ‖ ‗ ‘ ’ ‚ “ ” „ † ‡ • … ‰ ‱ ′ ″ ‴ ‵ ‶ ‷ ‹ › ※ ‽ ⁄ ⁒ ⁗

Currency Symbols: ₡ ₤ ₦ ₩ ₫ € ₱ ₲

Combining Diacritical Marks for Symbols: ⃐ ⃑ ⃒ ⃓ ⃔ ⃕ ⃖ ⃗ ⃘ ⃛ ⃜ ⃝ ⃞ ⃟ ⃠ ⃡ ⃤ ⃥ ⃦ ⃧ ⃨ ⃩ ⃪ ⃫ ⃬ ⃭ ⃮ ⃯ ⃰

Letterlike Symbols: ℂ ℃ ℇ ℉ ℊ ℋ ℌ ℍ ℎ ℏ ℐ ℑ ℒ ℓ ℕ № ℗ ℘ ℙ ℚ ℛ ℜ ℝ ℞ ℠ ™ ℤ Ω ℧ ℨ K Å ℬ ℭ ℮ ℯ ℰ ℱ ℳ ℴ ℵ ℶ ℷ ℸ ℼ ℽ ℾ ℿ ⅀ ⅅ ⅆ ⅇ ⅈ ⅉ

Arrows: ← ↑ → ↓ ↔ ↕ ↖ ↗ ↘ ↙ ↚ ↛ ↞ ↟ ↠ ↡ ↢ ↣ ↤ ↥ ↦ ↧ ↩ ↪ ↫ ↬ ↭ ↮ ↰ ↱ ↲ ↳ ↴ ↵ ↶ ↷ ↺ ↻ ↼ ↽ ↾ ↿ ⇀ ⇁ ⇂ ⇃ ⇄ ⇅ ⇆ ⇇ ⇈ ⇉ ⇊ ⇋ ⇌ ⇍ ⇎ ⇏ ⇐ ⇑ ⇒ ⇓ ⇔ ⇕ ⇖ ⇗ ⇘ ⇙ ⇚ ⇛ ⇜ ⇝ ⇦ ⇧ ⇨ ⇩ ⇳ ⇵ ⇶

Mathematical Operators: ∀ ∁ ∂ ∃ ∄ ∅ ∆ ∇ ∈ ∉ ∊ ∋ ∌ ∍ ∎ ∏ ∐ ∑ − ∓ ∔ ∕ ∖ ∗ ∘ ∙ √ ∝ ∞ ∟ ∠ ∡ ∢ ∣ ∤ ∥ ∦ ∧ ∨ ∩ ∪ ∫ ∬ ∭ ∮ ∯ ∰ ∱ ∲ ∳ ∴ ∵ ∶ ∷ ∸ ∹ ∺ ∻ ∼ ∽ ∾ ∿ ≀ ≁ ≂ ≃ ≄ ≅ ≆ ≇ ≈ ≉ ≊ ≋ ≌ ≍ ≎ ≏ ≐ ≑ ≒ ≓ ≔ ≕ ≖ ≗ ≘ ≙ ≚ ≛ ≜ ≝ ≞ ≟ ≠ ≡ ≢ ≣ ≤ ≥ ≦ ≧ ≨ ≩ ≪ ≫ ≬ ≭ ≮ ≯ ≰ ≱ ≲ ≳ ≴ ≵ ≶ ≷ ≸ ≹ ≺ ≻ ≼ ≽ ≾ ≿ ⊀ ⊁ ⊂ ⊃ ⊄ ⊅ ⊆ ⊇ ⊈ ⊉ ⊊ ⊋ ⊌ ⊍ ⊎ ⊏ ⊐ ⊑ ⊒ ⊓ ⊔ ⊕ ⊖ ⊗ ⊘ ⊙ ⊚ ⊛ ⊜ ⊝ ⊞ ⊟ ⊠ ⊡ ⊢ ⊣ ⊤ ⊥ ⊦ ⊧ ⊨ ⊩ ⊪ ⊫ ⊬ ⊭ ⊮ ⊯ ⊲ ⊳ ⊴ ⊵ ⊶ ⊷ ⊸ ⊹ ⊺ ⊻ ⊼ ⊽ ⊾ ⊿ ⋀ ⋁ ⋂ ⋃ ⋄ ⋅ ⋆ ⋇ ⋈ ⋉ ⋊ ⋋ ⋌ ⋍ ⋎ ⋏ ⋐ ⋑ ⋒ ⋓ ⋔ ⋕ ⋖ ⋗ ⋘ ⋙ ⋚ ⋛ ⋜ ⋝ ⋞ ⋟ ⋠ ⋡ ⋢ ⋣ ⋤ ⋥ ⋦ ⋧ ⋨ ⋩ ⋪ ⋫ ⋬ ⋭ ⋮ ⋯ ⋰ ⋱

Miscellaneous Technical: ⌀ ⌅ ⌆ ⌈ ⌉ ⌊ ⌋ ⌌ ⌍ ⌎ ⌏ ⌜ ⌝ ⌞ ⌟ ⌠ ⌡ ⌢ ⌣ 〈 〉 ⎛ ⎜ ⎝ ⎞ ⎟ ⎠ ⎡ ⎢ ⎣ ⎤ ⎥ ⎦ ⎧ ⎨ ⎩ ⎪ ⎫ ⎬ ⎭ ⎮ ⎰ ⎱ ⎲ ⎳ ⎴ ⎵ ⎶ ⏜ ⏝ ⏞ ⏟ ⏠ ⏡

Box Drawing, Block Elements: ─ │ ┌ ┐ └ ┘ ├ ┤ ┬ ┴ ┼ ▁ █ ░ ▒ ▓

Geometric Shapes: ■ □ ▪ ▫ ▬ ▭ ▲ △ ▶ ▷ ▼ ▽ ◀ ◁ ◊ ○ ● ◦ ◯

Miscellaneous Symbols, Dingbats: ♠ ♡ ♢ ♣ ♤ ♥ ♦ ♧ ♪ ♭ ♮ ♯ ⚭ ⚮ ⚲ ✓ ✠ ➡

Miscellaneous Mathematical Symbols-A: ⟂ ⟘ ⟙ ⟚ ⟛ ⟜ ⟝ ⟞ ⟠ ⟡ ⟢ ⟣ ⟦ ⟧ ⟨ ⟩ ⟪ ⟫ ⟮ ⟯

Supplemental Arrows-A, B: ⟴ ⟵ ⟶ ⟷ ⟸ ⟹ ⟺ ⟻ ⟼ ⟽ ⟾ ⟿ ⤆ ⤇

Supplemental Mathematical Operators: ⨀ ⨁ ⨂ ⨃ ⨄ ⨅ ⨆ ⨉ ⨌ ⨕ ⨯ ⨿ ⩽ ⩾ ⪅ ⪆ ⪇ ⪈ ⪉ ⪊ ⪋ ⪌ ⪕ ⪖ ⪯ ⪰

Miscellaneous Symbols and Arrows: ⬄ ⬅ ⬆ ⬇ ⬌ ⬍ ⬚ ⭅ ⭆ ⭋

CJK Symbols and Punctuation: 〚 〛 Alphabetic Presentation Forms: ﬀ ﬁ ﬂ ﬃ ﬄ

Mathematical Alphanumeric Symbols: **A B C D E F G H I J K L M N O P Q R S T U V W X Y Z a b c d e f g h i j k l m n o p q r s t u v w x y z** *A B C D E F G H I J K L M N O P Q R S T U V W X Y Z a b c d e f g i j k l m n o p q r s t u v w x y z* ***A B C D E F G H I J K L M N O P Q R S T U V W X Y Z a b c d e f g h i j k l m n o p q r s t u v w x y z*** 𝒜 𝒞 𝒟 𝒢 𝒥 𝒦 𝒩 𝒪 𝒫 𝒬 𝒮 𝒯 𝒰 𝒱 𝒲 𝒳 𝒴 𝒵 𝒶 𝒷 𝒸 𝒹 𝒻 𝒽 𝒾 𝒿 𝓀 𝓁 𝓂 𝓃 𝓅 𝓆 𝓇 𝓈 𝓉 𝓊 𝓋 𝓌 𝓍 𝓎 𝓏 𝓐 𝓑 𝓒 𝓓 𝓔 𝓕 𝓖 𝓗 𝓘 𝓙 𝓚 𝓛 𝓜 𝓝 𝓞 𝓟 𝓠 𝓡 𝓢 𝓣 𝓤 𝓥 𝓦 𝓧 𝓨 𝓩 𝓪 𝓫 𝓬 𝓭 𝓮 𝓯 𝓰 𝓱 𝓲 𝓳 𝓴 𝓵 𝓶 𝓷 𝓸 𝓹 𝓺 𝓻 𝓼 𝓽 𝓾 𝓿 𝔀 𝔁 𝔂 𝔃 𝔄 𝔅 𝔇 𝔈 𝔉 𝔊 𝔍 𝔎 𝔏 𝔐 𝔑 𝔒 𝔓 𝔔 𝔖 𝔗 𝔘 𝔙 𝔚 𝔛 𝔜 𝔞 𝔟 𝔠 𝔡 𝔢 𝔣 𝔤 𝔥 𝔦 𝔧 𝔨 𝔩 𝔪 𝔫 𝔬 𝔭 𝔮 𝔯 𝔰 𝔱 𝔲 𝔳 𝔴 𝔵 𝔶 𝔷 𝔸 𝔹 𝔻 𝔼 𝔽 𝔾 𝕀 𝕁 𝕂 𝕃 𝕄 𝕆 𝕊 𝕋 𝕌 𝕍 𝕎 𝕏 𝕐 𝕒 𝕓 𝕔 𝕕 𝕖 𝕗 𝕘 𝕙 𝕚 𝕛 𝕜 𝕝 𝕞 𝕟 𝕠 𝕡 𝕢 𝕣 𝕤 𝕥 𝕦 𝕧 𝕨 𝕩 𝕪 𝕫 𝕬 𝕭 𝕮 𝕯 𝕰 𝕱 𝕲 𝕳 𝕴 𝕵 𝕶 𝕷 𝕸 𝕹 𝕺 𝕻 𝕼 𝕽 𝕾 𝕿 𝖀 𝖁 𝖂 𝖃 𝖄 𝖅 𝖆 𝖇 𝖈 𝖉 𝖊 𝖋 𝖌 𝖍 𝖎 𝖏 𝖐 𝖑 𝖒 𝖓 𝖔 𝖕 𝖖 𝖗 𝖘 𝖙 𝖚 𝖛 𝖜 𝖝 𝖞 𝖟 𝖠 𝖡 𝖢 𝖣 𝖤 𝖥 𝖦 𝖧 𝖨 𝖩 𝖪 𝖫 𝖬 𝖭 𝖮 𝖯 𝖰 𝖱 𝖲 𝖳 𝖴 𝖵 𝖶 𝖷 𝖸 𝖹 𝖺 𝖻 𝖼 𝖽 𝖾 𝖿 𝗀 𝗁 𝗂 𝗃 𝗄 𝗅 𝗆 𝗇 𝗈 𝗉 𝗊 𝗋 𝗌 𝗍 𝗎 𝗏 𝗐 𝗑 𝗒 𝗓 𝗔 𝗕 𝗖 𝗗 𝗘 𝗙 𝗚 𝗛 𝗜 𝗝 𝗞 𝗟 𝗠 𝗡 𝗢 𝗣 𝗤 𝗥 𝗦 𝗧 𝗨 𝗩 𝗪 𝗫 𝗬 𝗭 𝗮 𝗯 𝗰 𝗱 𝗲 𝗳 𝗴 𝗵 𝗶 𝗷 𝗸 𝗹 𝗺 𝗻 𝗼 𝗽 𝗾 𝗿 𝘀 𝘁 𝘂 𝘃 𝘄 𝘅 𝘆 𝘇 𝘈 𝘉 𝘊 𝘋 𝘌 𝘍 𝘎 𝘏 𝘐 𝘑 𝘒 𝘓 𝘔 𝘕 𝘖 𝘗 𝘘 𝘙 𝘚 𝘛 𝘜 𝘝 𝘞 𝘟 𝘠 𝘡 𝘢 𝘣 𝘤 𝘥 𝘦 𝘧 𝘨 𝘩 𝘪 𝘫 𝘬 𝘭 𝘮 𝘯 𝘰 𝘱 𝘲 𝘳 𝘴 𝘵 𝘶 𝘷 𝘸 𝘹 𝘺 𝘻 𝘼 𝘽 𝘾 𝘿 𝙀 𝙁 𝙂 𝙃 𝙄 𝙅 𝙆 𝙇 𝙈 𝙉 𝙊 𝙋 𝙌 𝙍 𝙎 𝙏 𝙐 𝙑 𝙒 𝙓 𝙔 𝙕 𝙖 𝙗 𝙘 𝙙 𝙚 𝙛 𝙜 𝙝 𝙞 𝙟 𝙠 𝙡 𝙢 𝙣 𝙤 𝙥 𝙦 𝙧 𝙨 𝙩 𝙪 𝙫 𝙬 𝙭 𝙮 𝙯 𝙰 𝙱 𝙲 𝙳 𝙴 𝙵 𝙶 𝙷 𝙸 𝙹 𝙺 𝙻 𝙼 𝙽 𝙾 𝙿 𝚀 𝚁 𝚂 𝚃 𝚄 𝚅 𝚆 𝚇 𝚈 𝚉 𝚊 𝚋 𝚌 𝚍 𝚎 𝚏 𝚐 𝚑 𝚒 𝚓 𝚔 𝚕 𝚖 𝚗 𝚘 𝚙 𝚚 𝚛 𝚜 𝚝 𝚞 𝚟 𝚠 𝚡 𝚢 𝚣 𝚤 𝚥 𝚨 𝚩 𝚪 𝚫 𝚬 𝚭 𝚮 𝚯 𝚰 𝚱 𝚲 𝚳 𝚴 𝚵 𝚶 𝚷 𝚸 𝚹 𝚺 𝚻 𝚼 𝚽 𝚾 𝚿 𝛀 𝛁 𝛂 𝛃 𝛄 𝛅 𝛆 𝛇 𝛈 𝛉 𝛊 𝛋 𝛌 𝛍 𝛎 𝛏 𝛐 𝛑 𝛒 𝛓 𝛔 𝛕 𝛖 𝛗 𝛘 𝛙 𝛚 𝛛 𝛜 𝛝 𝛞 𝛟 𝛠 𝛡 𝛢 𝛣 𝛤 𝛥 𝛦 𝛧 𝛨 𝛩 𝛪 𝛫 𝛬 𝛭 𝛮 𝛯 𝛰 𝛱 𝛲 𝛳 𝛴 𝛵 𝛶 𝛷 𝛸 𝛹 𝛺 𝛻 𝛼 𝛽 𝛾 𝛿 𝜀 𝜁 𝜂 𝜃 𝜄 𝜅 𝜆 𝜇 𝜈 𝜉 𝜊 𝜋 𝜌 𝜍 𝜎 𝜏 𝜐 𝜑 𝜒 𝜓 𝜔 𝜕 𝜖 𝜗 𝜘 𝜙 𝜚 𝜛 𝜜 𝜝 𝜞 𝜟 𝜠 𝜡 𝜢 𝜣 𝜤 𝜥 𝜦 𝜧 𝜨 𝜩 𝜪 𝜫 𝜬 𝜭 𝜮 𝜯 𝜰 𝜱 𝜲 𝜳 𝜴 𝜵 𝜶 𝜷 𝜸 𝜹 𝜺 𝜻 𝜼 𝜽 𝜾 𝜿 𝝀 𝝁 𝝂 𝝃 𝝄 𝝅 𝝆 𝝇 𝝈 𝝉 𝝊 𝝋 𝝌 𝝍 𝝎 𝝏 𝝐 𝝑 𝝒 𝝓 𝝔 𝝕 𝝖 𝝗 𝝘 𝝙 𝝚 𝝛 𝝜 𝝝 𝝞 𝝟 𝝠 𝝡 𝝢 𝝣 𝝤 𝝥 𝝦 𝝧 𝝨 𝝩 𝝪 𝝫 𝝬 𝝭 𝝮 𝝯 𝝰 𝝱 𝝲 𝝳 𝝴 𝝵 𝝶 𝝷 𝝸 𝝹 𝝺 𝝻 𝝼 𝝽 𝝾 𝝿 𝞀 𝞁 𝞂 𝞃 𝞄 𝞅 𝞆 𝞇 𝞈 𝞉 𝞊 𝞋 𝞌 𝞍 𝞎 𝞏 𝞐 𝞑 𝞒 𝞓 𝞔 𝞕 𝞖 𝞗 𝞘 𝞙 𝞚 𝞛 𝞜 𝞝 𝞞 𝞟 𝞠 𝞡 𝞢 𝞣 𝞤 𝞥 𝞦 𝞧 𝞨 𝞩 𝞪 𝞫 𝞬 𝞭 𝞮 𝞯 𝞰 𝞱 𝞲 𝞳 𝞴 𝞵 𝞶 𝞷 𝞸 𝞹 𝞺 𝞻 𝞼 𝞽 𝞾 𝞿 𝟀 𝟁 𝟂 𝟃 𝟄 𝟅 𝟆 𝟇 𝟈 𝟉 𝟊 𝟋 𝟎 𝟏 𝟐 𝟑 𝟒 𝟓 𝟔 𝟕 𝟖 𝟗 𝟘 𝟙 𝟚 𝟛 𝟜 𝟝 𝟞 𝟟 𝟠 𝟡 𝟢 𝟣 𝟤 𝟥 𝟦 𝟧 𝟨 𝟩 𝟪 𝟫 𝟬 𝟭 𝟮 𝟯 𝟰 𝟱 𝟲 𝟳 𝟴 𝟵 𝟶 𝟷 𝟸 𝟹 𝟺 𝟻 𝟼 𝟽 𝟾 𝟿

Unicode: the typographer's friend or enemy?

An important subject, closely related to the contents and repertoire issues, is briefly mentioned in Sections 19, 28, and 34: the problem of the character set defined by the Unicode standard [35], and specified for math fonts in *Unicode Technical Report #25* [37].

The Unicode Consortium claims that »the Unicode standard follows a set of fundamental principles« and gives, among others, the following »principle«: *characters, not glyphs* and *semantics*. We are not able to reconcile these principles with the cases discussed in this section.

It should be emphasized that not all glyphs used extensively in typography, in particular, in or out of math formulas, are assigned Unicode numbers. Examples of such glyphs are small caps and old style numerals. Nothing in these two classes of glyphs has received Unicode numbers, although there are codes for much more narrowly used double struck characters or monospaced and sans-serif digits.

Another example of »unicodeless« glyphs are the pieces used for assembling extensible characters (cf. page 28, Figures 5 and 6).

It is not so bad if whole blocks of characters are included or excluded. The worse situation is an inconsistency of including only some glyphs. The abovementioned double struck glyphs are a good example of such a situation. Here is the group of double struck glyphs assigned Unicode slots, without apparent rhyme or reason:

```
2102 DOUBLE-STRUCK CAPITAL C
210D DOUBLE-STRUCK CAPITAL H
2115 DOUBLE-STRUCK CAPITAL N
2119 DOUBLE-STRUCK CAPITAL P
211A DOUBLE-STRUCK CAPITAL Q
211D DOUBLE-STRUCK CAPITAL R
2124 DOUBLE-STRUCK CAPITAL Z
213C DOUBLE-STRUCK SMALL PI
213D DOUBLE-STRUCK SMALL GAMMA
213E DOUBLE-STRUCK CAPITAL GAMMA
213F DOUBLE-STRUCK CAPITAL PI
2140 DOUBLE-STRUCK N-ARY SUMMATION
2145 DOUBLE-STRUCK ITALIC CAPITAL D
2146 DOUBLE-STRUCK ITALIC SMALL D
2147 DOUBLE-STRUCK ITALIC SMALL E
2148 DOUBLE-STRUCK ITALIC SMALL I
2149 DOUBLE-STRUCK ITALIC SMALL J
```

The remaining letters of the alphabet (upper and lower case) and digits received mathematical codes (1D538–1D56B), with gaps at the glyphs above.

Another notable example are sub- and superscripts, theoretically not needed in math fonts, because the sub- and superscript characters (»unicodeless«) are accessed there by the OTF feature mechanism (the ssty feature, cf. page 28). In practice, for legacy reasons, sub- and superscript glyphs are expected to be present in a text font, and, consequently, in the basic (plain) charset of a math font too—see Table 1. The Unicode standard [35] enumerates the following Latin letters in this context:

```
1D62 LATIN SUBSCRIPT SMALL LETTER I
1D63 LATIN SUBSCRIPT SMALL LETTER R
1D64 LATIN SUBSCRIPT SMALL LETTER U
1D65 LATIN SUBSCRIPT SMALL LETTER V
2090 LATIN SUBSCRIPT SMALL LETTER A
2091 LATIN SUBSCRIPT SMALL LETTER E
2092 LATIN SUBSCRIPT SMALL LETTER O
2093 LATIN SUBSCRIPT SMALL LETTER X
2094 LATIN SUBSCRIPT SMALL LETTER SCHWA
2095 LATIN SUBSCRIPT SMALL LETTER H
2096 LATIN SUBSCRIPT SMALL LETTER K
2097 LATIN SUBSCRIPT SMALL LETTER L
2098 LATIN SUBSCRIPT SMALL LETTER M
2099 LATIN SUBSCRIPT SMALL LETTER N
209A LATIN SUBSCRIPT SMALL LETTER P
209B LATIN SUBSCRIPT SMALL LETTER S
209C LATIN SUBSCRIPT SMALL LETTER T
2C7C LATIN SUBSCRIPT SMALL LETTER J@
2071 SUPERSCRIPT LATIN SMALL LETTER I
207F SUPERSCRIPT LATIN SMALL LETTER N
```

Besides the somewhat surprising presence of the ›schwa‹ character and the striking asymmetry between the number of sub- and superscripts, most questionable here is the incompleteness of the Latin alphabet. So far, we have not included these glyphs in neither text nor math fonts.

Another example concerns math italic symbols. The Unicode standard reads:

```
1D44E MATHEMATICAL ITALIC SMALL A
1D44F MATHEMATICAL ITALIC SMALL B
1D450 MATHEMATICAL ITALIC SMALL C
1D451 MATHEMATICAL ITALIC SMALL D
1D452 MATHEMATICAL ITALIC SMALL E
1D453 MATHEMATICAL ITALIC SMALL F
1D454 MATHEMATICAL ITALIC SMALL G
                                ⟸?
1D456 MATHEMATICAL ITALIC SMALL I
1D457 MATHEMATICAL ITALIC SMALL J
1D458 MATHEMATICAL ITALIC SMALL K
```

```
1D459 MATHEMATICAL ITALIC SMALL L
1D45A MATHEMATICAL ITALIC SMALL M
1D45B MATHEMATICAL ITALIC SMALL N
1D45C MATHEMATICAL ITALIC SMALL O
1D45D MATHEMATICAL ITALIC SMALL P
1D45E MATHEMATICAL ITALIC SMALL Q
1D45F MATHEMATICAL ITALIC SMALL R
1D460 MATHEMATICAL ITALIC SMALL S
1D461 MATHEMATICAL ITALIC SMALL T
1D462 MATHEMATICAL ITALIC SMALL U
1D463 MATHEMATICAL ITALIC SMALL V
1D464 MATHEMATICAL ITALIC SMALL W
1D465 MATHEMATICAL ITALIC SMALL X
1D466 MATHEMATICAL ITALIC SMALL Y
1D467 MATHEMATICAL ITALIC SMALL Z
```

Math italic ›h‹ is apparently missing. In fact, the Unicode Consortium decided to leave the slot

1D455 unused forever and »inflate« the meaning of the slot formerly assigned to the Planck constant:

```
210E PLANCK CONSTANT
     = height, specific enthalpy, ...
     * simply a mathematical italic h;
       this character's name results
       from legacy usage
```

More on Unicode's ambiguities, inconsistencies, riddles, curiosities, etc. concerning typography applications, especially math typesetting, can be found in Piotr Strzelczyk's ruminations [23].

From the typographer's point of view, the enumeration of all signs used in the world does not seem to be a good idea. Therefore, it should be no surprise that it turned out to be impossible to create a math OTF font that has an internally logical and coherent structure and, at the same time, is practically useful. Conforming rigorously to the mentioned specifications does not help too much—it would lead, in our opinion, to fonts containing mostly seldom used glyphs. Needless to say, research examining which characters from the Unicode repertoire are actually used in practice, would be welcome. Lacking such empirical data, we adopted Cambria Math as our »reference point«. Cambria Math contains circa 6500 glyphs; we decided to reduce this number to at most 4500 for the TEX Gyre series of math fonts. We can reluctantly consider proposals for suitable extensions, if truly needed.

Compatibility issues

We already explained why 100-percent compatibility of the LM text fonts with the parent Computer Modern fonts is unfeasible; thereby, the same applies, even to a greater extent, to the LM Math font (cf. the case of horizontal extensible braces in page 28).

Recall that we used an alternative calligraphic alphabet in LM Math (page 28). This incompatibility can be relatively easily patched, by including two calligraphic scripts in the font and using OTF ›stylistic sets‹ features, implemented as the OTF features `ss01`–`ss20` (see [33]).

There is, however, a flaw shared by both Computer Modern and Euler calligraphic alphabets and thus inherited by the Latin Modern Math: the lack of a corresponding lower case alphabet. The Unicode specification assigns slots to lower case mathematical calligraphic letters, implying that they are expected to be present in math fonts.

We could not find a calligraphic font with a proper license optically matching Euler or Computer Modern upper case calligraphic letters, and we gave up in advance the idea of designing the respective matching lower case letters ourselves. Instead, we began to consider the inclusion of yet another stylistic set, a »home-made« calligraphic font inspired by (but not based on) the original Computer Modern calligraphic script. Although we would never dare to make a text calligraphic font without close cooperation by a professional type designer, we took a chance and attempted to prepare a symbol calligraphic font for TG DejaVu Math.

There was an additional reason for doing our own calligraphic script. Anticipating future work (see page 42 below), we looked for a calligraphic script matching a sans-serif font. Having not found any, we decided to prepare our own. Because our calligraphic alphabet for TG DejaVu Math was, of course, defined by a parametric MetaType1 program, it was possible to prepare a sans-serif (linear) variant of the calligraphic script with reasonable effort. A tentative, experimental linear calligraphic alphabet is shown in Figure 9: the calligraphic script used in TG DejaVu Math (top) and its linear variant to be used in a sans-serif math font (bottom).

Fortunately, there is no issue of compatibility regarding the TG math fonts, as there are no predecessors. The only question is the similarity of repertoire between the LM and TG math fonts. As explained in page 34, the repertoires are bound to differ slightly, for the usual legacy reasons.

For the same reasons, some technical details of the font structure also cannot be implemented similarly. Worthy of mentioning in this context is a peculiar difference between the LM and TG math fonts related to integrals; namely, the TG math fonts

Figure 9

contain extensible integrals, which are definable within the OTF math format—but we are not aware of any typesetting engine that can take advantage of this possibility.

Plans for the future

Testing and maintenance

Tasks that are important today and will be forever important in the future are maintenance and testing. There is, of course, neither a single tool for testing nor a unique maintenance procedure. Each case demands a specific approach.

It is Piotr Pianowski who is responsible for testing fonts and preparing adequate tools. Tests refer to both the appearance of the fonts and their internal structure. In particular, the intermediate PostScript Type 1 code needs checking. The following example shows a case where the inspection of the code revealed an error in the `parenright.ex` procedure (describing an extender of the extensible right parenthesis, which should be just a rectangle), probably due to the wrong rounding procedure: the number ›1‹ means that the line drawn by the command `rlineto` is not strictly vertical. The left column contains the correct code for the corresponding component of the extensible left parenthesis:

```
/parenleft.ex {      /parenright.ex {
  143 609 hsbw         338 609 hsbw
  127 418 rmoveto      128 418 rmoveto
  -127 hlineto         -128 hlineto
  -418 vlineto         1 -418 rlineto
  127 hlineto          126 hlineto
  closepath            closepath
  endchar              endchar
  } ND                 } ND
```

It is next to impossible to perceive such a tiny deformity on a printout of glyph shapes, which does not mean that the bug should not be fixed.

The next example, Figure 10, shows one of the tests we use for checking a vexing problem regarding Greek letter names. Some Greek letters have a shape variant, and there is an unfortunate discrepancy between TEXies and the rest of the world (the rightmost two columns marked with asterisks) as to which glyphs are considered »normal« and which »variant«.

Figure 10

texgyredejavu-math.otf

	normal	var	normal*	var*
\mstd	$\epsilon\,\phi\,\rho\,\theta$	$\varepsilon\,\varphi\,\varrho\,\vartheta$	$\varepsilon\,\varphi\,\rho\,\theta$	$\epsilon\,\phi\,\varrho\,\vartheta$
\mrm	$\epsilon\,\varphi\,\rho\,\theta$	$\varepsilon\,\phi\,\varrho\,\vartheta$	$\varepsilon\,\phi\,\rho\,\theta$	$\epsilon\,\varphi\,\varrho\,\vartheta$
\mit	$\epsilon\,\phi\,\rho\,\theta$	$\varepsilon\,\varphi\,\varrho\,\vartheta$	$\varepsilon\,\varphi\,\rho\,\theta$	$\epsilon\,\phi\,\varrho\,\vartheta$
\mbf	$\boldsymbol{\epsilon\,\varphi\,\rho\,\theta}$	$\boldsymbol{\varepsilon\,\phi\,\varrho\,\vartheta}$	$\boldsymbol{\varepsilon\,\phi\,\rho\,\theta}$	$\boldsymbol{\epsilon\,\varphi\,\varrho\,\vartheta}$
\mbi	$\boldsymbol{\epsilon\,\phi\,\rho\,\theta}$	$\boldsymbol{\varepsilon\,\varphi\,\varrho\,\vartheta}$	$\boldsymbol{\varepsilon\,\varphi\,\rho\,\theta}$	$\boldsymbol{\epsilon\,\phi\,\varrho\,\vartheta}$
\mbfss	$\boldsymbol{\epsilon\,\phi\,\rho\,\theta}$	$\boldsymbol{\varepsilon\,\varphi\,\varrho\,\vartheta}$	$\boldsymbol{\varepsilon\,\varphi\,\rho\,\theta}$	$\boldsymbol{\epsilon\,\phi\,\varrho\,\vartheta}$
\mitss	$\epsilon\,\phi\,\rho\,\theta$	$\varepsilon\,\varphi\,\varrho\,\vartheta$	$\varepsilon\,\varphi\,\rho\,\theta$	$\epsilon\,\phi\,\varrho\,\vartheta$
\mbiss	$\boldsymbol{\epsilon\,\phi\,\rho\,\theta}$	$\boldsymbol{\varepsilon\,\varphi\,\varrho\,\vartheta}$	$\boldsymbol{\varepsilon\,\varphi\,\rho\,\theta}$	$\boldsymbol{\epsilon\,\phi\,\varrho\,\vartheta}$

Almost every time we deal with the Greek alphabet, a mistake in variant names tries to creep in. Without tests of this kind we would be lost.

The last example concerning maintenance and testing issues, Figure 11, shows a typical test of the structure of extensible characters, »embellished« horizontal arrows in this case: ›vh 2‹ means that there are 2 size variants, ›ch 3‹ and ›ch 5‹—that there are 3 and 5 horizontal components of a given glyph, respectively. Tests of this kind are essential for maintaining uniformity across a font collection as well as inside a single font.

As a result of remarks to date from the users of our fonts, we gathered a list of recommended fixes and improvements—some trivial, like reports on malformed

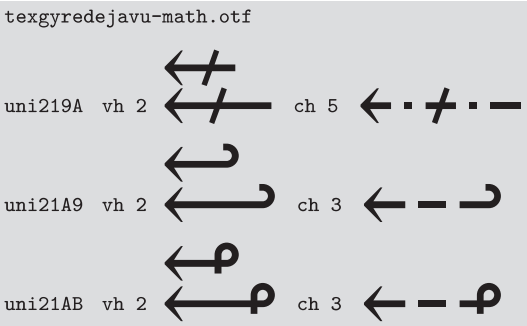


Figure 11

glyphs or wrongly assigned Unicode slots, some fairly difficult, like suggestions to implement anchors or math staircase kerns.

The latter two potential improvements are still pending, mainly because of vague specification and the question of practical application. Being unsure whether all relevant typesetting programs would be able to handle such improvements, we preferred to linger till the engines would become stable. It seems that the time is ripe to attempt these extensions, especially as the staircase kerns were ultimately implemented in XeTeX in the middle of 2014.

We are also not sure which OTF features should be present in math fonts. At the moment, only math-oriented OTF features are implemented. Perhaps we should consider the inclusion of text-oriented OTF features too, like the mentioned onum, lnum, pnum, and tnum for switching between numeral variants, or stylistic sets for switching between calligraphic alphabets.

Also as suggested by users, we consider excerpting a number of glyphs, mostly geometrical shapes and arrows but also selected math operators and relational symbols, etc. (but excluding extensible and subscript characters), from math fonts and transferring them into the respective text fonts. Such glyphs, though prepared for math-oriented applications, can also be fruitfully used in conventional fonts, that is, those not equipped with the MATH table, for the typesetting of technical documents. This, obviously, requires a proper specification and careful selection of the glyphs in question. This is, in fact, one of the planned stages of work on new fonts.

Of course, MetaType1 itself also requires maintenance: enhancing, modifying, fixing, etc. For example, we had to extend the otherwise stable set of METAPOST macros used in MetaType1 in order to handle the math extension of the OTF specification.

Commencing our works for the GUST e-foundry, we underestimated the inexorable Hofstadter's Law (see footnote 7 on page 28) which apparently applies not only to

time resources. We believed that MetaType1 could be kept simple: just METAPOST, a few trivial Gawk scripts and a stand-alone, stable converter to PostScript Type 1 fonts, T1utils, and that's all. In accordance with Hofstadter's Law, the task has unavoidably turned out to be much more complex than we expected and the implementation—too heterogeneous.

In order to remedy this, we intend to unify MetaType 1 by eliminating Gawk, Perl, T1utils, and, as we mentioned in page 16, AFDKO. We aim at employing two basic »subengines«, namely, METAPOST (for generating outlines) and Python (for arranging the METAPOST output) plus one external, possibly replaceable, »subengine«, for now, the FontForge Python library (for generating OTF fonts and the theoretically obsolescent but still widely used PostScript Type 1 fonts).

More GUST e-foundry fonts

In the nearest future, we would like to elaborate and release three experimental math fonts, namely: bold math, sans-serif math, and monospaced math. These fonts fit neither the Microsoft nor Unicode specifications ([29] and [37], respectively). Therefore, we have to start by working out an altered specification, adjusted to our purposes, for these non-standard cases. For example, it is not at all clear what to do with the sans-serif alphabet in a sans-serif math font: should it be omitted, left intact, or modified somehow (how?).

Bold math fonts can be used in titles containing math formulas, as shows Figure 12 (a tentative version of TG Termes Bold Math is used).

Figure 12

1. Equivalence of the integral $\left(\oiint_{\partial S} \mathbf{E} \cdot d\mathbf{S} = 0\right)$ and differential $(\nabla \cdot \mathbf{E} = 0)$ formulas

In this section, we shall prove that the integral and differential forms of Maxwell's equation known as "Gauss's law for magnetism", i.e.:

$$\oiint_{\partial S} \mathbf{E} \cdot d\mathbf{S} = 0 \qquad (1)$$

and

$$\nabla \cdot \mathbf{E} = 0 \qquad (2)$$

are equivalent.

Nowadays, many documents are being typeset in sans-serif, even schoolbooks. Computer presentations may serve as another example. For such applications sans-

serif math seems plausible. An example of such an application is shown in Figure 13 (a tentative version of TG DejaVu Sans Math is used).

Figure 13

Various notations for continued fractions

The integers a_0, a_1 etc., are called *coefficients* or *terms* of the continued fraction. One can abbreviate the continued fraction

$$a_0 + \cfrac{1}{a_1 + \cfrac{1}{a_2 + \cfrac{1}{a_3}}}$$

in the notation of Carl Friedrich Gauss as

$$x = a_0 + \mathop{\mathbf{K}}_{i=1}^{3} \frac{1}{a_i}$$

or in the notation of Alfred Pringsheim as

$$x = a_0 + \frac{1\,|}{|a_1} + \frac{1\,|}{|a_2} + \frac{1\,|}{|a_3}.$$

source: http://en.wikipedia.org/wiki/Continued_fraction

TEX users are accustomed to the use of control sequences for math-oriented glyphs such as `\infty`, `\sum` or `\pm`. Some of these glyphs can be accessed (typed in and displayed) directly in text editors, provided the font used by the editor contains the relevant glyphs. The lion's share of nominally math-oriented glyphs can also be prepared as monospaced glyphs, usable in text editors, provided they are assigned Unicode numbers. One can expect that it will improve the legibility of sources and, thereby, the efficiency of preparation of such documents—see Figure 14 (a tentative version of TG DejaVu Mono Math is used).

Note that treating subscripts incoherently by the Unicode Standard (page 38) may prove to be a significant impediment in the latter case, that is, for fonts without full subscript support. Perhaps the defining of the complete set of subscripts and superscripts (partially »unicodeless«) and accessing them via stylistic set features (`ss01`–`ss20`) can be a solution, provided a given text editor handles the relevant OTF features.

Legal issues

The problem of copyrights and licenses has clung to us like a leech from the very beginning and still persists. We are not copyright experts and we do not want to be. Being sick of trouble with releasing our work due to discussions about legal

```
\section{Алгоритм де Кастельє}

Заданий многочлен Бернштейна $B$ ступеня $n$
з опорними точками $β₀,…,βₙ$
$$
  B(t) = ∑ⁿᵢ₌₀ βᵢb_{i,n}(t),
$$
де $b$ — базис многочлена Бернштейна, многочлен
в точці $t₀$ може бути визначений за допомогою
рекурентного співвідношення:
$$\eqalign{
  βᵢ⁽⁰⁾& = βᵢ \mbox{, } i=0,…,n\cr
  βᵢ^{(j)}& = βᵢ^{(j-1)}(1-t₀) + βᵢ₊₁^{(j-1)}t₀
     \mbox{, } i=0,…,n-j \mbox{, } j=1,…,n\cr
}$$
Тоді визначення $B$ в точці $t₀$ може бути
визначено в $n$ кроків алгоритму.
Результат $B(t₀)$ дано за:
$$
  B(t₀)=β₀⁽ⁿ⁾.
$$
Також, крива Безьє $B$ може бути розділена
в точці $t₀$ на дві криві з відповідними
опорними точками:
$$
  β₀⁽⁰⁾,β₀⁽¹⁾,…,β₀⁽ⁿ⁾
  β₀⁽ⁿ⁾,β₁⁽ⁿ⁻¹⁾,…,βₙ⁽⁰⁾
$$
```

Figure 14

source: https://uk.wikipedia.org/wiki/Алгоритм_де_Кастельє

aspects of distributing free fonts, we coined a pun *lice-sense*. Just one example: the release of TG DejaVu Math was delayed by about a year because of doubts raised concerning legal matters.

Having said this, we should emphasize that it does not mean that there has been no activity from the side of the GUST e-foundry regarding legal issues. On the contrary, our magnificent liaison officer, Jerzy Ludwichowski, has made Herculean efforts in order to provide appropriate licenses for GUST fonts. In particular, he prepared a proposal of a license for the URW fonts that would suit GUST e-foundry needs, managed to contact in person the URW++ managing director, Dr. Peter Rosenfeld, and, after long negotiations, received the gracious approval for the additional license.

All the fonts released so far are licensed under the GUST Font License (GFL; see [30]), except for TG DejaVu Math which has a somewhat complex license (see [9], the Manifest file for TG DejaVu Math). Recently, many fonts are being released under the SIL Open Font License (OFL; see [34]); therefore, we consider dual-licensing GUST fonts (GFL+OFL).

More details concerning legal matters relevant to GUST e-foundry fonts can be found in a series of publications by Jerzy Ludwichowski—see, for example [18, 19, 20, 21].

Constraints of tradition vs. our dreams

We are not particularly enthusiastic about font technology nowadays, but we are not going to spit into the wind, and try to make the best of what is available. This does not mean, however, that we are going to relinquish dreams of a successor to the currently prevalent »Knuth–Gutenberg« model of typesetting, that is, the model of stiff rectangles (types) arranged within a larger rectangle (page; a series of pages being called a document), deeply ingrained in Johannes Gutenberg's technology of movable type, and transferred by Donald E. Knuth, among others, to the realm of computers.

Computers facilitate some operations, such as kerning and ligatures, thereby accelerating work on documents. The ease of use of affine transformations is also considered an advantage of computers by many graphic designers. We are inclined to consider both these aspects as being of rather equivocal benefit. Leaving aside these philosophical questions and a far-reaching yet obvious idea, in fact also philosophical, that a font could be defined as a structured collection of general purpose object programs, we confine ourselves to pointing out two examples of aspects that might be improved within the framework of the contemporary edifice of typesetting.

Shrinkable and stretchable spaces, as in TEX, would supposedly be convenient also in OTF fonts and seem not too hard to implement. This simple problem touches on a fairly general and not in the least trivial problem which could be called »the conflict of competence«: which »knowledge« should be implemented in a font and which—in a typesetting program.

Another improvement, this time hard to implement, is related to a naïve question: why must additional alphabets be embedded into a single math font? The answer is simple: operating systems are poorly designed with regard to font management. In particular, a user is not allowed to define a »private« family of fonts—the commonplace pattern »regular, regular italic, bold, bold italic« is very difficult to dislodge. One can imagine other variants of this idea, namely, borrowing components for extensible characters or kerns from several fonts. As far as we know, such an approach has been implemented neither in any typesetting program nor in any operating system. TEX is no exception (unless virtual fonts are used); note, for example, that changing fonts within a word switches off the TEX hyphenation mechanism.

It does not seem as if amendments of this kind are to be introduced in any near future. It is understandable, as the constraints of tradition and compatibility usually slow down the innovative processes. For example, recently announced advancements in the OTF format specification (see, e.g. [6]), can actually be considered a step backward towards a previously abandoned idea, temporarily as it turns out, of

multiple master fonts. Anyway, this announcement augurs both ill and well for us: it means, on the one hand, that we will probably have to reimplement MetaType1 in order to keep pace with the surrounding world and, on the other hand, that we still have something to do and something to think about.

Acknowledgements

We are indebted to all the people and TEX groups that have supported our font enterprises. Almost all the GUST e-foundry projects have been kindly supported by the Czechoslovak TEX user group CSTUG, the German-speaking TEX user group DANTE e.V., the Polish TEX Users Group GUST, the Dutch-speaking TEX user group NTG, TUG India, UK-TUG, and, last but not least, TUG. In a few cases, GUTenberg, the French-speaking TEX Users Group, supported us too.

We are very grateful to Karel Píška and Ulrik Vieth, who performed extensive tests of our fonts and inspired us with insightful comments, and to Marcin Woliński, who provided the indispensable LATEX support for our fonts. The exceptional, personal thanks we owe to our friends who kept our spirits up for many years and tirelessly encouraged us to work on fonts: Hans Hagen, Johannes Küster, Jurek Ludwichowski, Volker RW Schaa, Jola Szelatyńska—hearty thanks! All trademarks belong to their respective owners and have been used here for informational purposes only.

Literature and Software

[1] Lars Engebretsen, *Almost European Fonts*
https://www.ctan.org/tex-archive/fonts/ae

[2] Michael Everson, *The Alphabets of Europe*
http://www.evertype.com/alphabets/

[3] Lee Hetherington, Eddie Kohler, *T1utils*
http://www.lcdf.org/type/t1asm.1.html
http://www.lcdf.org/type/t1disasm.1.html

[4] John D. Hobby, *METAPOST*
https://ctan.org/pkg/metapost

[5] Donald E. Knuth, *The TEXbook*, *TEX: The Program*, *The METAFONTbook*, *METAFONT: The Program*, *Computer Modern Typefaces*, Computers & Typesetting, vol. A–E, Addison-Wesley, Reading, Massachusetts, 1986

[6] John Hudson, *Introducing OpenType Variable Fonts*, 2016
https://medium.com/@tiro/https-medium-com-tiro-introducing-opentype-variable-fonts-12ba6cd2369

[7] Bogusław Jackowski, Janusz M. Nowacki, Piotr Strzelczyk, *Antykwa Półtawskiego: a parameterized outline font*, Proceedings of the EuroTEX Conference, Heidelberg, Germany, 1999
article: http://www.staff.uni-giessen.de/partosch/eurotex99/jackowski
download: http://www.gust.org.pl/projects/e-foundry/poltawski

[8] Bogusław Jackowski, Marek Ryćko, *Polish extension of Computer Modern fonts*
https://www.ctan.org/pkg/pl-mf

[9] Bogusław Jackowski, Piotr Strzelczyk, Piotr Pianowski, GUST e-foundry math fonts:
The Latin Modern Math (LM Math) font
http://www.gust.org.pl/projects/e-foundry/lm-math
The TEX Gyre (TG) Math Fonts
http://www.gust.org.pl/projects/e-foundry/tg-math
https://www.ctan.org/tex-archive/fonts/tex-gyre-math

[10] Bogusław Jackowski, Piotr Strzelczyk, *How to make more than one math OpenType font or the Beasts of Fonts*, DANTE 2011 Meeting, Bremen, Germany, 2011
http://www.gust.org.pl/projects/e-foundry/math/beasts05.pdf

[11] Bogusław Jackowski, Janusz M. Nowacki, Piotr Strzelczyk, *The Latin Modern (LM) Family of Fonts*
www.gust.org.pl/projects/e-foundry/latin-modern

[12] Bogusław Jackowski, Janusz M. Nowacki, *Latin Modern fonts: how less means more*
Proceedings of the EuroTEX Conference, Pont-à-Mousson, France, 2005,
http://www.tug.org/TUGboat/Articles/tb27-0/jackowski.pdf

[13] Bogusław Jackowski, Janusz M. Nowacki, Piotr Strzelczyk, *MetaType 1: a METAPOST-based engine for generating Type 1 fonts*, 2001
article: http://www.ntg.nl/maps/26/15.pdf
presentation: http://ntg.nl/eurotex/JackowskiMT.pdf
download: http://www.ctan.org/tex-archive/fonts/utilities/metatype1/

[14] Bogusław Jackowski, Janusz M. Nowacki, Piotr Strzelczyk, *The TEX Gyre (TG) Collection of Fonts*
http://www.gust.org.pl/projects/e-foundry/tex-gyre

[15] Bogusław Jackowski, Janusz M. Nowacki, Piotr Strzelczyk, *TEX Gyre Pagella Math or Misfortunes of Math Typographer*, BachoTEX XX, Bachotek, Poland, 2012
http://www.gust.org.pl/projects/e-foundry/math/misfortunes02.pdf

[16] Jörg Knappen, Norbert Schwartz, *European Computer Modern Fonts*
https://www.ctan.org/pkg/ec

[17] Donald E. Knuth, *METAFONT and METAPOST logo fonts*
https://www.ctan.org/tex-archive/fonts/mflogo

[18] Jerzy B. Ludwichowski, *GUST font licenses*, BachoTEX XIV, Bachotek, Poland, 2006
https://www.tug.org/fonts/licenses/gfl.pdf

[19] Jerzy B. Ludwichowski, Karl Berry, *GUST Font License: An application of the LaTeX Project Public License*, XVII European TEX Conference and BachoTEX XV, Bachotek, Poland, 2007; TUGboat, Volume 29 (2008), No. 1
http://www.gust.org.pl/projects/e-foundry/licenses/tb91Berry_Ludwichowski.pdf

[20] Jerzy B. Ludwichowski, *Licensing of the TEX Gyre family of fonts*, XIX European TEX Conference and 3rd International ConTEXt User Meeting, The Hague, The Netherlands, 2009
https://www.ntg.nl/EuroTeX/2009/slides/jerzy-slides.pdf

[21] Jerzy B. Ludwichowski, *Is there life besides licensing?*, DANTE e.V. General Meeting, Bremen, Germany, 2011
https://www.dante.de/events/Archiv/dante2011/programm/vortraege/folien-jl.pdf

[22] Janusz M. Nowacki, *Polish fonts*
http://jmn.pl/en/

[23] Piotr Strzelczyk, *Standard Unicode w typografii*, Acta Poligraphica nr 1/2013 (in Polish; to be published also in English under the title *Standard Unicode in typography*)
http://www.cobrpp.com.pl/actapoligraphica/uploads/pdf/AP2013_01_Strzelczyk.pdf
see also: Piotr Strzelczyk *(uni)coding of math fonts*, BachoTEX XIX, Bachotek, Poland, 2011
http://www.gust.org.pl/bachotex/2011-en/presentations/Strzelczyk_1_2011

[24] Ulrik Vieth, *OpenType math illuminated*, TUGboat, Volume 30 (2009), No. 1
http://www.tug.org/tugboat/tb30-1/tb94vieth.pdf

[25] George Williams and the FontForge Project contributors, *FontForge*
http://fontforge.github.io/en-US/

[26] *Adobe Font Development Kit for OpenType*
http://www.adobe.com/devnet/opentype/afdko.html

[27] *Adobe Core 35 fonts*

Adobe original AFM files: ftp://ftp.adobe.com/pub/adobe/type/win/all/afmfiles/base35/
URW replacement: http://www.ctan.org/pkg/urw-base35

[28] *Adobe Type 1 Font Format*
https://partners.adobe.com/public/developer/en/font/T1_SPEC.PDF

[29] *MATH – The mathematical typesetting table*
https://www.microsoft.com/typography/OTSPEC/math.htm

[30] *GUST Font License*
http://www.gust.org.pl/fonts/licenses/GUST-FONT-LICENSE.txt
http://tug.org/fonts/licenses/GUST-FONT-LICENSE.txt

[31] *OpenType specification (full)*
http://www.microsoft.com/en-ph/download/details.aspx?id=1144

[32] *OpenType Feature File Specification*
http://www.adobe.com/devnet/opentype/afdko/topic_feature_file_syntax.html

[33] *Registered features—definitions and implementations (p–t)*
https://www.microsoft.com/typography/otspec/featuretags.htm

[34] *SIL Open Font License*
https://opensource.org/licenses/OFL-1.1

[35] *The Unicode Standard 9.0.0, 2016*
http://unicode.org/Public/UNIDATA/NamesList.txt

[36] *The Unicode Standard: A Technical Introduction*
http://www.unicode.org/standard/principles.html

[37] Barbara Beeton, Asmus Freytag, Murray Sargent III, *Unicode Technical Report #25. Unicode Support for Mathematics*
http://www.unicode.org/reports/tr25/

All the links above were accessed 05.07.2015.

Der Satz kritischer Editionen mit LaTeX und reledmac

Philipp Pilhofer

Der druckfertige Satz einer wissenschaftlich-kritischen Edition ist eine komplexe Aufgabe. In diesem Artikel wird ein Weg vorgestellt, den Anforderungen mit LaTeX unter Benutzung der Pakete reledmac und reledpar auf höchstem Niveau zu entsprechen, und dabei gleichzeitig eine flexible Arbeitsweise zu ermöglichen. Das offene Dateiformat erlaubt eine reibungslose Weiterbearbeitung, etwa für eine digitale Edition.

Der Satz einer kritischen Edition

Im Rahmen meines Promotionsprojektes arbeite ich an einer kritischen Edition eines spätantiken griechischen Textes mit einer deutschen Übersetzung. Eine solche Edition stellt hohe Anforderungen an das Satzprogramm, insbesondere dann, wenn ein doppelseitiger Satz angestrebt wird: Auf den jeweils linken Seiten soll der Originaltext stehen, auf den rechten Seiten die Übersetzung, die paragraphenweise auf gleicher Höhe wie links stehen muss. Hinzu kommen links und rechts mehrere voneinander unabhängige Fußnotenapparate, eine durchgehende Zeilenzählung u. v. m. Die Apparate auf den linken Seiten haben verschiedene Aufgaben, in meinem Fall sind dies:

1. Der Zeugen-Apparat: Dieser zeigt zeilengenau an, auf Basis welcher Handschriften die entsprechenden Bereiche des Textes erstellt wurden.
2. Der Quellen-Apparat: In diesem Apparat wird auf Parallelstellen in anderen Texten verwiesen, die dem Autor/der Autorin möglicherweise als Vorlage dienten, oder auf die er/sie anspielt.
3. Der textkritische Apparat: Dieser ist das Kernstück der kritischen Edition. Hier werden Text-Varianten der einzelnen Handschriften angezeigt, die von dem oben gedruckten Text abweichen. Dabei wird die Zeile und das Wort (oder die Wörter) aus dem Haupttext (das »Lemma«) angegeben, dahinter die abweichenden Varianten mit den jeweiligen Handschriften.

In Zeiten des Bleisatzes wurden einfach so viele Korrekturdurchgänge absolviert, bis das satztechnische Ergebnis zufriedenstellend war. In den Zeiten des beginnenden Computersatzes wurde die Lage unübersichtlicher, es gibt Berichte der Benutzung von Microsoft Word unter Zuhilfenahme von Schere und Kleber. [8, S. 34] Heute wird meistens ein Word-ähnliches Programm namens Classical Text Editor (CTE) verwendet, das die verschiedenen Apparate in vielen kleinen Fenstern organisiert. Der CTE erfüllt also die technischen Grundbedingungen, dennoch liegen die Nachteile auf der Hand: Die entstehenden Dateien liegen in einem binären Format vor und können nur

mit dem CTE bearbeitet werden. Man ist also für alle Zeiten auf einen lauffähigen CTE angewiesen, die Entwicklung des proprietären Codes hängt jedoch seit 20 Jahren an einer einzigen Person. Eine spätere Weiterverarbeitung der druckfertigen Edition ist aus den binären Dateien fast nicht möglich; für eine digitale Edition beispielsweise müsste man wieder ganz von vorne beginnen. In Zeiten schmaler werdender Budgets sind auch die Lizenzkosten nicht zu vernachlässigen.

Bei der Nutzung von LaTeX stellen sich diese Probleme nicht, Darüber hinaus bieten sich weitere Vorteile wie der überlegene Textsatz. Mit ednotes und reledmac liegen zwei Pakete vor, die allen Anforderungen einer kritischen Edition gerecht werden.[1] reledmac befindet sich seit Ende der 80er Jahren in mehr oder weniger steter Fortentwicklung. Der aktuelle Maintainer Maïeul Rouquette arbeitet Fehlerberichte oder Feature-Wünsche freundlich und effizient ab.[2] Das Paket hat sich in mehreren publizierten Editions(groß)projekten als gleichermaßen stabil und flexibel erwiesen. Beispielsweise verwendet die Erlanger Athanasius-Arbeitsstelle seit zehn Jahren (re)ledmac und hat damit bereits mehrere Bände publiziert, unter anderem [1]. Zu den technischen und insbesondere TeXnischen Hintergründen vgl. [8]. Eine unvollständige Liste der Editionen in den unterschiedlichsten Sprachen, die auf (rel)edmac basieren, findet sich unter [4]. reledmac ist ausführlich dokumentiert, die verschiedenen Funktionen werden auf 70 Seiten erläutert; zudem liegen dem Paket 40 Beispieldateien bei, die viele der Konfigurationsmöglichkeiten vorführen. [6]

Das Setzen einer Edition mit LaTeX und reledmac

Auf den folgenden Seiten will ich nicht alle dieser Funktionen vorstellen, sondern nur eine kurze Einführung geben und eine mögliche Grundkonfiguration für eine den Anforderungen entsprechende kritische Edition vorstellen. Davon ausgehend sollte es dann einfach sein, eventuelle Sonderwünsche mit Hilfe der Dokumentation umzusetzen.

Das Paket wird mit `\usepackage[<opt>]{reledmac}` geladen. Da reledmac ein sehr mächtiges Paket ist, kann es den TeX-Durchlauf spürbar verlangsamen. Je nach Anforderungsszenario und verwendeter Hardware kann es sinnvoll sein, die nicht benötigten Funktionen des Paketes abzuschalten. Standardmäßig ermöglicht reledmac je fünf Endnoten-, »kritische« Fußnoten- und »normale« Fußnoten-Apparate (jeweils

[1] Auf das Paket ednotes [2] von Uwe Lück gehe ich hier nicht weiter ein, ein ausführlicher *TUGboat*-Artikel liegt diesem Paket bei. Eine Arbeit, die mit diesem Paket erstellt wurde, ist [3].

[2] Das Paket reledmac [6] geht ursprünglich auf das plain TeX-Paket edmac von John Lavagnino und Dominik Wujastyk zurück. Dieses wurde ab 2003 von Peter Wilson für LaTeX als ledmac portiert und weiterentwickelt. Im Jahr 2011 hat Maïeul Rouquette das Paket übernommen und erst als eledmac, später als reledmac fortgeführt. Als Beispiel für die Effizienz mag hier erwähnt sein, dass ein Fehler im Paket reledmac, der beim Schreiben dieses Artikels aufgefallen ist, ganze zwölf Minuten nach Absenden des Bug-Reports gefixt war.

mit A bis E bezeichnet). Es lassen sich auch weitere Apparate hinzufügen. Mit »normalen« Fußnoten bezeichne ich die üblichen Fußnoten, die oben im Text eine Zahl setzen und unten hinter der Zahl den Text der Fußnote. Bei einer »kritischen« Fußnote findet sich keine Markierung im Text, dafür wird unten eine Zeilenangabe gemacht und das Lemma wiederholt, bevor der Text der Fußnote folgt. In unserem Beispiel gehe ich nur auf die Fußnoten-Apparate ein, und wie oben erwähnt, brauchen wir nur drei dieser Apparate, der Rest (inklusive einiger weiterer hier nicht besprochener Möglichkeiten) wird also abgeschaltet:

```
\usepackage[%
  series={A,B,C},% nur die Apparate A B C aktivieren
  noend,        % keine Endotenapparate
  noeledsec,    % keine eledsections et al.
  noledgroup    % keine ledgroups
]{reledmac}
```

Der Text der Edition muss zwingend von \beginnumbering … \endnumbering eingefasst werden. Jeder Paragraph muss mit \pstart begonnen und mit \pend beendet werden. Alternativ kann dies mit \autopar vereinfacht werden. [6, S. 17] Die Paragraphen-Nummer kann man automatisch ausgeben lassen, wenn man dies möchte.[3] Hier ein Beispiel der bisher eingeführten Befehle:[4]

```
\beginnumbering
\pstart[\subsubsection{Strabons Geographika XIV 5,1}]
Τῆς Κιλικίας δὲ τῆς ἔξω τοῦ Ταύρου ἡ μὲν λέγεται τραχεῖα ἡ δὲ πεδιάς;
τραχεῖα μέν, ἧς ἡ παραλία στενή ἐστι καὶ οὐδὲν ἢ σπανίως ἔχει \dots{}
\pend
\endnumbering
```

Das Ergebnis sähe bisher folgendermaßen aus:

Strabons Geographika XIV 5,1

Τῆς Κιλικίας δὲ τῆς ἔξω τοῦ Ταύρου ἡ μὲν λέγεται τραχεῖα ἡ δὲ πεδιάς; τραχεῖα μέν, ἧς ἡ παραλία στενή ἐστι καὶ οὐδὲν ἢ σπανίως ἔχει …

[3] Dafür gibt es die Befehle \numberpstarttrue und \sidepstartnumtrue. Der erste fügt zu Beginn des Paragraphen die Nummer ein, der zweite schreibt die Nummer an den Rand und schaltet den sichtbaren Zeilenzähler ab.

[4] Der Beispieltext ist Strabons Geographika XIV 5,1. Text und Übersetzung folgen dabei mit Abweichungen der Radtschen Ausgabe. [5, S. 96 f.] Die hier verwendeten Text-Varianten, Handschriftenzeugen und Quellen-Belege sind rein fiktiv und nur gewählt, um die Funktionsweise des Paketes reledmac einfach zu veranschaulichen.

Die Zeilen zwischen den numbering-Befehlen werden fortlaufend gezählt. Schon bei der Zeilenzählung gibt es viele Einstellungsmöglichkeiten. Mit dem Befehl \lineation{<arg>} lässt sich erreichen, dass der Zeilenzähler auf jeder Seite (page) oder mit jedem Paragraphen (pstart) zurückgesetzt wird. Normalerweise wird der Stand des Zeilenzählers in jeder fünften Zeile angezeigt. Dies lässt sich mit \linenumincrement{<num>} ändern: Um die Funktionalität auf möglichst wenig Raum erklären zu können, wollen wir hier mit \linenumincrement{2} den Zählerstand in jeder zweiten Zeile anzeigen lassen. Mit \firstlinenum{2} wird eingestellt, dass der Zeilenzähler in der 2. Zeile auch zum ersten Mal steht.

Für Zwischenüberschriften, die keine Zeilennummern (und also auch keinen Apparat) erhalten sollen, kann man die üblichen Befehle verwenden: \chapter, \section usw. Zu beachten ist dabei, dass diese als optionales Argument des \pstart-Befehles mitgegeben werden müssen. Möglich sind auch Überschriften mit Zeilenzählung und Apparat, dazu sind die Befehle \eledchapter, \eledsection, usw. zu verwenden (diese können innerhalb eines Paragraphen ohne Beschränkungen verwendet werden).[5]

Neben dem Text sind Randbemerkungen durch verschiedene Befehle möglich, beispielsweise über \ledsidenote{<anm>}. Diese Randbemerkungen lassen sich etwa dazu verwenden, den Seitenumbruch einer älteren Edition desselben Textes anzuzeigen. Dazu definiert man am besten einen Befehl, der einerseits an der richtigen Stelle der Zeile ein entsprechendes Zeichen (hier: |) setzt und gleichzeitig am Rand die neue Seite ausgibt. Durch einen eingebauten Zähler wird die ausgegebene Seitenzahl bei jeder Benutzung automatisch um eins erhöht. Dieses Beispiel gibt am Rand den Buchstaben T (der für die alte Edition steht) mit der Seitenzahl 269 aus:

```
\newcounter{alteSeite}
\setcounter{alteSeite}{269}
\newcommand\alteSeite{{|\ledsidenote{\emph{T}~\thealteSeite\stepcounter{alteSeite}}}}
```

Mit den nun neu eingeführten Befehlen sieht der Beispielcode so aus:

```
\firstlinenum{2}% ab der zweiten Zeile die Nummern anzeigen
\linenumincrement{2}% ... und zwar jede zweite Zeile
\beginnumbering
\pstart[\subsubsection{Strabons Geographika XIV 5,1}]
Τῆς Κιλικίας δὲ τῆς ἔξω τοῦ Ταύρου ἡ μὲν λέγεται τραχεῖα ἡ δὲ πεδιάς;
τραχεῖα μέν, ἧς ἡ παραλία \alteSeite{} στενή ἐστι καὶ οὐδὲν ἢ σπανίως ἔχει \dots{}
\pend
\endnumbering
```

[5] Da wir diese Art von Überschriften im vorliegenden Beispiel nicht brauchen, haben wir sie in der Präambel mit noeledsec abgeschaltet.

Das Ergebnis hat sich damit auch verändert und sähe nun folgendermaßen aus:

Strabons Geographika XIV 5,1

Τῆς Κιλικίας δὲ τῆς ἔξω τοῦ Ταύρου ἡ μὲν λέγεται τραχεῖα ἡ δὲ πεδιάς;
2 τραχεῖα μέν, ἧς ἡ παραλία | στενή ἐστι καὶ οὐδὲν ἢ σπανίως ἔχει … T 269

Ich hatte oben drei Apparate erwähnt, die wir für unser Beispiel verwenden wollen: Zeugen-Apparat, Quellen-Apparat und textkritischer Apparat (siehe Seite 53). Die letzten beiden haben eine ähnliche Funktionsweise: Ein Wort (oder einige Wörter) aus dem Text, das Lemma, soll unten im Apparat mit Zeilenangabe wiedergegeben werden, dahinter folgt eine Art von Kommentar oder Anmerkung zum Lemma. Der erstgenannte Apparat hingegen soll zeilenweise angeben, bei welchen Handschriften (den »Zeugen«) die jeweiligen Zeilen nachgewiesen werden können.

Ich gehe zuerst auf den Quellen-Apparat und den textkritischen Apparat ein. Diese Apparate lassen sich mit dem Befehl `\edtext{<lemma>}{<befehl>}` ansprechen. `lemma` ist dabei das Wort (oder die Wörter), zu dem im Apparat eine Anmerkung gemacht werden soll:[6] Das `lemma` wird einmal normal oben im Text ausgegeben und einmal im Apparat mit Zeilenangabe. Bei `befehl` ist einzutragen, in welchen Apparat die Anmerkung mit dem `lemma` aufgenommen werden soll, dazu existieren die Befehle `\Afootnote{<anm>}`, `\Bfootnote{<anm>}` usw. Für den textkritischen Apparat, der ganz unten stehen soll, wählen wir Apparat C. Eine textkritische Bemerkung zum Wort Κιλικίας sähe also so aus, wenn die Handschrift V dort Πισιδίας liest (es ist zu beachten, dass `reledmac` bis zu drei LaTeX-Durchläufe braucht, bis die Zeilennummern im Apparat richtig sind):

```
Τῆς \edtext{Κιλικίας}{\Afootnote{Πισιδίας V}} δὲ τῆς ἔξω τοῦ Ταύρου \dots{}
```

Wenn das `lemma` etwas länger ist, kann es sinnvoll sein, es nicht vollständig im Apparat wiederzugeben, sondern es stattdessen abzukürzen. Dafür existiert der Befehl `\lemma{<lemma_kurz>}`, der auch im `befehl`-Feld von `\edtext` zu verwenden ist. Die Verwendung ist auch dann sinnvoll, wenn der Text des Lemmas andere Befehle enthält, die unten im Apparat nicht noch einmal wiederholt werden sollen, wie `\alteSeite` oder verschachtelte `\edtext`-Befehle. Hier ein Anwendungsbeispiel (unten auf Seite 60 wird dies auch noch anschaulich gemacht):

[6] In dem Fall, dass ein Wort mehrfach innerhalb einer Zeile vorkommt, ist der Befehl `\sameword{<lemma>}` anzuwenden. Es ist zu beachten, dass der unten eingeführte Befehl `\vari` auf den Befehl `\lemma` zurückgreift: Dieser Tatsache muss bei der Benutzung von `\sameword` Rechnung getragen werden. Näheres ist der Dokumentation von `reledmac` zu entnehmen.

```
\dots{}, ἥς \edtext{ἡ παραλία \alteSeite{} στενή ἐστι}{\lemma{ἡ \dots{}
ἐστι}\Bfootnote{Anmerkungstext}} καὶ οὐδὲν ἧ σπανίως ἔχει \dots{}
```

Da dies schnell etwas unübersichtlich wird, führen wir einen Befehl für Text-Varianten ein, nämlich: \vari[<lemma_kurz>]{<lemma>}{<anm>} (die Angabe des abgekürzten Lemmas ist optional). Dieser erledigt alle genannten Aufgaben auf einmal:

```
\newcommand\vari[3][]{%
  \edtext{#2}{\if$#1$ #2\else\lemma{#1}\fi
    \Cfootnote{#3}}}
```

Der neue Befehl sieht im Beispieltext dann so aus:

```
Τῆς \vari{Κιλικίας}{Πισιδίας V} δὲ τῆς ἔξω τοῦ Ταύρου \dots{}
```

Der angegebene Code führt nach genügend vielen Durchläufen zu diesem Ergebnis:

Strabons Geographika XIV 5,1

Τῆς Κιλικίας δὲ τῆς ἔξω τοῦ Ταύρου …

1 Κιλικίας] Πισιδίας V

Für den Quellenapparat wählen wir Apparat B und definieren entsprechend den Befehl \quell[<lemma_kurz>]{<lemma>}{<anm>}:

```
\newcommand\quell[3][]{%
  \edtext{#2}{\if$#1$ #2\else\lemma{#1}\fi
    \Bfootnote{#3}}}
```

Gehen wir hier einmal davon aus, dass die Wörter ἡ παραλία στενή ἐστι schon bei einem fiktiven älteren Autor namens Konon Paraliotes stehen, in seinem ebenso fiktiven Werk Κοσμογραφία VI 7. Dann würde der Beispielcode so aussehen (nun auch unter Verwendung des abgekürzten Lemmas und zudem mit einer weiteren Lesart: Handschrift G bietet das ἔξω nicht):

```
Τῆς \vari{Κιλικίας}{Πισιδίας V} δὲ τῆς \vari{ἔξω}{\emph{del.} G}
τοῦ Ταύρου ἡ μὲν λέγεται τραχεῖα ἡ δὲ πεδιάς;
τραχεῖα μέν, ἥς \quell[ἡ \dots{} ἐστι]{ἡ παραλία \alteSeite{} στενή ἐστι}%
{Konon Paraliotes, Κοσμογραφία VI 7} καὶ οὐδὲν ἧ σπανίως ἔχει \dots{}
```

Mit beiden Apparaten kommen wir zu diesem Zwischenergebnis:

Strabons Geographika XIV 5,1

Τῆς Κιλικίας δὲ τῆς ἔξω τοῦ Ταύρου ἡ μὲν λέγεται τραχεῖα ἡ δὲ πεδιάς;
2 τραχεῖα μέν, ἧς ἡ παραλία | στενή ἐστι καὶ οὐδὲν ἢ σπανίως ἔχει … T 269

2 ἡ … ἐστι] Konon Paraliotes, Κοσμογραφία VI 7

1 Κιλικίας] Πισιδίας V
1 ἔξω] del. G

Jetzt benötigen wir nur noch den Zeugen-Apparat. Dieser funktioniert etwas anders als die beiden bisher beschriebenen Apparate. Standardmäßig verwendet reledmac dafür den Apparat A.[7] Mit dem Befehl \msdata{<zeugen>} werden die ab dieser Stelle herangezogenen Handschriften angegeben. Diese Angabe ist gültig bis zur nächsten Angabe über \msdata{<zeugen>} oder bis zu einem \stopmsdata.

Nehmen wir an, unser Text sei in insgesamt vier Handschriften V, U, G und T belegt; in V und T sind nur die ersten Zeilen erhalten, in U und G der komplette Text. Dann sähe unser Beispiel so aus:

```
\msdata{G T U V}Τῆς \vari{Κιλικίας}{Πισιδίας V} δὲ τῆς \vari{ἔξω}{\emph{del.} G}
τοῦ Ταύρου ἡ μὲν λέγεται τραχεῖα ἡ δὲ πεδιάς;
τραχεῖα μέν, ἧς \quell[ἡ \dots{} ἐστι]{ἡ παραλία \alteSeite{} στενή ἐστι}%
{Konon Paraliotes, Κοσμογραφία VI 7} καὶ οὐδὲν
\vari{ἢ σπανίως}{ἢ σπανιαίτερον V U; \emph{del.} T} ἔχει τι χωρίον
\msdata{G U}ἐπίπεδον, καὶ ἔτι ἧς ὑπέρκειται ὁ Ταῦρος οἰκούμενος κακῶς μέχρι καὶ τῶν
προσβόρρων πλευρῶν τῶν περὶ Ἴσαυρα καὶ τοὺς Ὁμοναδέας μέχρι τῆς Πισιδίας;
```

Noch nicht sonderlich schön, aber immerhin technisch vollständig, sieht der Zwischenstand so aus, wie auf der nächsten Seite gezeigt.

Nun wollen wir noch einige optische Feinheiten anpassen und die Anmerkungen platzsparender organisieren. Ich stelle hier einige der Möglichkeiten vor. Es kommen nun zahlreiche verschiedene Befehle vor, die aufgrund der vielen Einstellungsmöglichkeiten etwas verwirrend erscheinen mögen. Man sollte sich jedoch vor Augen halten, dass man diese Einstellungen nur einmal vornehmen muss, und sobald alles nach Wunsch aussieht, muss man daran nichts mehr ändern. Muss aber doch noch

[7] Dies lässt sich über \setmsdataseries{<app>} ändern.

Strabons Geographika XIV 5,1

Τῆς Κιλικίας δὲ τῆς ἔξω τοῦ Ταύρου ἡ μὲν λέγεται τραχεῖα ἡ δὲ
2 πεδιάς; τραχεῖα μέν, ἧς ἡ παραλία | στενή ἐστι καὶ οὐδὲν ἢ σπανίως T 269
ἔχει τι χωρίον ἐπίπεδον, καὶ ἔτι ἧς ὑπέρκειται ὁ Ταῦρος οἰκούμενος
4 κακῶς μέχρι καὶ τῶν προσβόρρων πλευρῶν τῶν περὶ Ἴσαυρα καὶ τοὺς
Ὁμοναδέας μέχρι τῆς Πισιδίας;

1–3 Ms.] G T U V

3–5 Ms.] G U

---

2 ἡ … ἐστι] Konon Paraliotes, Κοσμογραφία VI 7

---

1 Κιλικίας] Πισιδίας V

1 ἔξω] del. G

2 ἢ σπανίως] ἢ σπανιαίτερον V U; del. T

etwas geändert werden, genügt eine einzige Korrektur an einem dieser Befehle, um das ganze Dokument umzustellen.

Fast alle der hier vorgestellten Befehle haben ein optionales Argument: Wenn es leer bleibt, gilt der Befehl für alle Apparate; wenn man dort einen oder mehrere Apparate mit ihrem Buchstaben angibt, gilt der Befehl nur für diese. Die Anordnung der Anmerkungen im Apparat wird mit dem Makro `\Xarrangement[<app>]{<option>}` angepasst. Ich wähle hier die platzsparende Option `paragraph`, es wären aber auch zwei- oder dreispaltiger Satz der Anmerkungen möglich: `twocol`, `threecol`. Der Befehl muss vor dem `\beginnumbering` stehen.

Im gezeigten Beispiel sind in Z. 1 mehrere Varianten zu besprechen; die Zeilennummer wird im Moment einfach wiederholt. Schöner ist es, wenn stattdessen nur ein Trennsymbol angezeigt wird. Setzt man `\Xnumberonlyfirstinline[<app>][<bool>]` auf `true`, ist die doppelte Zeilennummer weg. Dann muss auch noch `\Xnumberonlyfirstintwolines[<app>][<bool>]` auf `true` gesetzt werden, damit bei mehrzeiligen Lemmata auf jeden Fall die Zeilennummern angezeigt werden. Mit `\Xsymlinenum{<symbol>}` kann noch ein Trennsymbol definiert werden, in unserem Fall ||. Um die Übersichtlichkeit der Apparate zu steigern, lassen wir durch den Befehl `\Xnotenumfont[<app>]{<stil>}` die Zeilennummern etwas dicker setzen. Mit `\Xnonbreakableafternumber[<app>]` verhindern wir, dass im Apparat nach der Angabe der Zeilennummer die Zeile umgebrochen wird.

Für den Fall, dass ein Lemma über zwei Zeilen geht, soll nicht beispielsweise »2–3« ausgegeben werden, sondern »2 f.«. Dies legen wir mit `\Xtwolines[<app>]{<abk>}` fest. Bei Lemmata, die über drei oder mehr Zeilen gehen, sollen die Zeilennummern aber vollständig ausgegeben werden, daher benutzen wir: `\Xtwolinesbutnotmore[<app>]`.

Mit `\Xlemmaseparator[<app>]{<sep>}` kann eingestellt werden, was direkt hinter dem Lemma ausgegeben werden soll, die Standard-Einstellung ist]. Für den Quellen-Apparat und den textkritischen Apparat soll es stattdessen ein »:« sein.

Dann nehmen wir noch einige Einstellungen für den Sonderfall des Zeugen-Apparates vor. Bisher steht hier hinter den Zeilennummern »Ms.«, dann der Lemmaseparator und dann die Zeugen. Hinter den Zeilennummern sollen aber nur die Zeugen stehen, weiter nichts. Mit `\setmsdatalabel{<label>}` streichen wir das »Ms.« vor jeder Anmerkung; allerdings erscheint nun am Anfang des Apparates ein »Ms.« Dieses entfernen wir mit dem Befehl `\Xtxtbeforenotes[<app>]{<label>}`. Mit `\Xafternumber[<app>]{<laenge>}` setzen wir das Spatium nach der Zeilennummer auf 2pt, mit dem gerade schon verwendeten Befehl `\Xlemmaseparator` löschen wir den Lemmaseparator. Damit statt des Lemmaseparators kein Spatium erscheint, setzen wir `\Xinplaceoflemmaseparator[<app>]{<laenge>}` auf 0 pt.

Auf Seiten, die durchgehend durch dieselben Handschriften bezeugt sind, stehen keine Zeilenangaben, sondern nur diese Handschriften. Bei den aktuellen Einstellungen wäre davor ein überflüssiges Spatium, das sich mit `\Xinplaceofnumber[<app>]{<laenge>}` noch entfernen lässt.

Damit haben wir folgende neue Optionen gesetzt:

```
%% generell zu den Apparaten
% auf jeden Fall vor \beginnumbering:
\Xarrangement{paragraph}%
% kann auch nach \beginnumbering stehen:
\Xnumberonlyfirstinline[][true]%
\Xnumberonlyfirstintwolines[][true]%
\Xsymlinenum{||}%
\Xnotenumfont[]{\bfseries}%
\Xnonbreakableafternumber[]%
\Xtwolines{f.}%
\Xtwolinesbutnotmore%

%% Quellen- und textkritischer Apparat
\Xlemmaseparator[B,C]{:}

%% Zeugen-Apparat:
```

```
\setmsdatalabel{}%
\Xtxtbeforenotes[A]{}%
\Xafternumber[A]{2pt}%
\Xlemmaseparator[A]{}%
\Xinplaceoflemmaseparator[A]{0pt}%
\Xinplaceofnumber[A]{0pt}%
```

Bei diesen Optionen sieht das Ergebnis nun so aus:

Strabons Geographika XIV 5,1

Τῆς Κιλικίας δὲ τῆς ἔξω τοῦ Ταύρου ἡ μὲν λέγεται τραχεῖα ἡ δὲ
2 πεδιάς; τραχεῖα μέν, ἧς ἡ παραλία | στενή ἐστι καὶ οὐδὲν ἢ σπανίως T 269
ἔχει τι χωρίον ἐπίπεδον, καὶ ἔτι ἧς ὑπέρκειται ὁ Ταῦρος οἰκούμενος
4 κακῶς μέχρι καὶ τῶν προσβόρρων πλευρῶν τῶν περὶ Ἴσαυρα καὶ τοὺς
Ὁμοναδέας μέχρι τῆς Πισιδίας;

1–3 G T U V **3–5** G U

2 ἡ … ἐστι: Konon Paraliotes, Κοσμογραφία VI 7

1 Κιλικίας: Πισιδίας V || ἔξω: del. G **2** ἢ σπανίως: ἢ σπανιαίτερον V U; del. T

Zum Abschluss sei hier noch auf die Befehle für Querverweise hingewiesen. Mit `\edlabel{<label>}` lässt sich ein Label definieren, das dann etwa mit den Befehlen `\edpageref{<label>}`, `\edlineref{<label>}` und `\pstartref{<label>}` angesprochen werden kann: Man kann sich also die Seite, die Zeile oder die Paragraphennummer eines Labels ausgeben lassen; dies ist eine sehr praktische Funktion, solange der Seiten- und Zeilenumbruch noch nicht feststeht. Eine Reihe weiterer Befehle ermöglicht automatische Querverweise auf Anmerkungen im Apparat, Zeilenenden u. v. m.

Der zweiseitige Satz einer Edition mit reledpar

Die Edition des griechischen Textes ist damit voll funktionsfähig. Als letztes wollen wir nun noch dafür sorgen, dass der Edition eine Übersetzung gegenübergestellt wird. Auf den jeweils linken Seiten soll der griechische Text mit seinen Apparaten stehen, auf den jeweils rechten die Übersetzung mit kommentierenden Anmerkungen. Für diesen Zweck gibt es das Paket reledpar, das gebündelt mit reledmac auf

CTAN verfügbar ist.[8] Diese beiden Pakete bauen aufeinander auf, daher können alle Funktionen im parallelen Satz auf beiden Seiten genutzt werden. Für viele reledmac-Befehle liegt nun ein analoger Befehl vor, um dieselben Einstellungen auf den rechten Seiten vornehmen zu können.

Für den parallelen Satz muss nicht viel getan werden. In eine pages-Umgebung müssen eine Leftside- und eine Rightside-Umgebung eingefügt werden. Innerhalb dieser kann die Edition so eingefügt werden, wie wir sie bisher erstellt haben. Nach der pages-Umgebung werden mit dem Befehl \Pages alle Seiten ausgegeben.

```
\begin{pages}
\begin{Leftside}
\beginnumbering
%... der griechische Text
\endnumbering
\end{Leftside}

\begin{Rightside}
\beginnumbering
%... der deutsche Text
\endnumbering
\end{Rightside}
\end{pages}

% hier nun alles ausgeben:
\Pages
```

Standardmäßig werden die Paragraphen jeweils auf derselben Höhe dargestellt, wobei wieder mehrere LaTeX-Durchläufe benötigt werden. Weitere Einstellungen sind fast nicht nötig. Mit den Befehlen \firstlinenumR{<num>} und \linenumincrementR{<num>} stellen wir den rechten Zeilenzähler analog zum linken ein.

Auf den rechten Seiten benötigen wir keine Anmerkungen, die jeweils auf die Zeile(n) und ein Lemma verweisen. Stattdessen wollen wir nur »normale« Fußnoten verwenden, die im Text eine hochgestellte Zahl setzen. Der übliche Befehl \footnote sollte zwar funktionieren; möchte man jedoch auch hier mehrere Apparate verwenden, sollten die Befehle \footnoteA, \footnoteB usw. verwendet werden. Diese dürfen nicht mit den Befehlen \Afootnote, \Bfootnote usw., verwechselt werden, die wir für die Apparate auf den linken Seiten verwendet haben (und die nur innerhalb eines \edtext funktionieren). Prinzipiell können beide Arten von Fußnoten links und

[8] Siehe [7]. Das Paket unterstützt auch paralellen Satz in zwei Spalten. Die Befehle funktionieren analog zu den hier vorgestellten zum zweiseitigen Satz.

rechts verwendet werden, aber – wie gesagt – brauchen wir rechts nur die normalen Fußnoten, die wir links wiederum nicht brauchen. Wie die gerade erklärten Einstellungen und Befehle eingegeben werden müssen, sollte mittlerweile deutlich geworden sein. Auf Seite 66 sieht man das fertige Ergebnis.

Weitere Vorteile der Nutzung von TeX

Wie eingangs erwähnt, liegen die TeX-Dateien als normale Textdateien vor. Dadurch sind sie leicht anschlussfähig an weitere Programme und ermöglichen so eine größere Flexibilität. Auf zwei dieser Möglichkeiten möchte ich hier kurz aufmerksam machen. Das eine sind digitale Editionen über XML-TEI. Bei einem heute in Angriff genommenen Editionsprojekt ist es sicherlich sinnvoll, neben der mit TeX erstellten Druckfassung auch die Option einer digitalen Edition offen zu halten. Bei der Verwendung von TeX mit reledmac ist eine Konvertierung der Dateien in ein XML-TEI-kompatibles Format ohne größeren Aufwand möglich, wenn man einige Konventionen beachtet. [8, S. 36–41] Damit steht der einfache Weg in die Digital Humanities offen.

Ein anderer Punkt ist die Verwendung von Versionsverwaltungssystemen wie subversion.[9] Eine solche Arbeitsweise mag in naturwissenschaftlichen Fächern vorausgesetzt werden, in den Geisteswissenschaften ist sie fast völlig unbekannt. Die Verwendung einer Versionsverwaltungssoftware mit externem Repositorium hat schon für Einzelpersonen große Vorteile: Durch die Versionsverwaltung hat man eine sichere Backup-Lösung und ein Archiv in einem, jede gespeicherte Version kann problemlos wiederhergestellt werden. Außerdem ist die Arbeit an mehreren PCs, etwa im Büro, zu Hause und am Laptop unterwegs unfallfrei möglich: Es kann nicht mehr vorkommen, dass man aus Versehen an einer veralteten Datei weiterarbeitet. Noch größer sind die Vorteile bei der Nutzung innerhalb von größeren Projekten mit mehreren Mitarbeitenden: Auf diese Weise können nicht nur mehrere Personen gleichzeitig an einem Projekt arbeiten, sondern auch mögliche Bearbeitungsfehler (über die blame-Funktion) schnell ausfindig gemacht und zurückgesetzt werden (revert).

Zusammenfassung

Das hier vorgestellte Beispieldokument lässt sich zu Übungszwecken vollständig herunterladen: http://kilikien.de/reledmac. Es ist mit XeLaTeX oder LuaLaTeX lauffähig; reledmac selbst ist aber auch mit pdfLaTeX verwendbar.

[9] Zur Verwendung von LaTeX mit subversion und Anwendungsbeispielen vgl. bereits [9].

Es sollte deutlich geworden sein, dass der Satz einer wissenschaftlich-kritischen Edition eine schwierige Aufgabe ist. Die Verwendung von LaTeX und reledmac ermöglicht einen vergleichsweise einfachen und nachhaltigen Weg zu einer solchen Edition. Ich habe in diesem Artikel anhand von vielen Beispielen gezeigt, welche Schritte es benötigt, um eine Edition mit Zeilenzählung, Zeugen-Apparat, Quellen-Apparat und textkritischem Apparat zu setzen und zwar parallel mit einer Übersetzung sowie einigen weiteren Feinheiten. Das Ergebnis ist eine Datei, die nur darauf wartet, zwischen zwei Buchdeckel gedruckt zu werden.

Literatur und Software

[1] Athanasius Werke III 4, Dritter Band, erster Teil: Dokumente zur Geschichte des Arianischen Streites, 4. Lieferung: Bis zur Synode von Alexandrien 362, (Hrsg.: Hanns Christof Brennecke u. a.), de Gruyter, Berlin und Boston, 2014.

[2] Uwe Lück: ednotes – Typeset scholarly editions, Version 1.3a, 2006, CTAN:/macros/latex/contrib/ednotes (besucht am 25. 11. 2016).

[3] Ioannis Antiocheni Fragmenta Quae Supersunt Omnia, (Hrsg.: Sergei Mariev), de Gruyter, Berlin und New York, 2008.

[4] Publizierte kritische Editionen, die (rel)edmac benutzt haben, https://www.zotero.org/groups/critical_editions_typeset_with_edmac_ledmac_and_eledmac/items (besucht am 25. 11. 2016).

[5] Strabons Geographika, Mit Übersetzung und Kommentar, Buch XIV–XVII: Text und Übersetzung, *Bd. 4*, (Hrsg.: Stefan Radt), Vandenhoeck & Ruprecht, Göttingen, 2005.

[6] Maïeul Rouquette: reledmac – Typeset scholarly editions with LaTeX, Version 2.17.2, 2017, CTAN:/macros/latex/contrib/reledmac (besucht am 26. 01. 2017).

[7] — reledpar – Typeset scholarly editions in parallel texts, Version 2.16.2, 2017, CTAN:/macros/latex/contrib/reledpar (besucht am 26. 01. 2017).

[8] Annette von Stockhausen: »Zwischen zwei Welten, Ein Langzeitprojekt in Zeiten des digitalen Umbruchs«, Magazin für digitale Editionswissenschaften, 2 (2016), 33–41.

[9] Uwe Ziegenhagen: »Dokumentenmanagement mit LaTeX und Subversion«, Die TeXnische Komödie, 3 (2008), 36–47.

Strabons Geographika XIV 5,1

Τῆς Κιλικίας δὲ τῆς ἔξω τοῦ Ταύρου ἡ μὲν λέγεται τραχεῖα ἡ δὲ πεδιάς; τραχεῖα μέν, ἧς ἡ παραλία | στενή ἐστι καὶ *T* 269
οὐδὲν ἢ σπανίως ἔχει τι χωρίον ἐπίπεδον, καὶ ἔτι ἧς ὑπέρκειται ὁ Ταῦρος οἰκούμενος κακῶς μέχρι καὶ τῶν προσβόρρων πλευρῶν τῶν περὶ Ἴσαυρα καὶ τοὺς Ὁμοναδέας μέχρι τῆς Πισιδίας;

πεδιὰς δ' ἡ ἀπὸ Σόλων καὶ Ταρσοῦ μέχρι Ἰσσοῦ, καὶ ἔτι ἧς ὑπέρκεινται κατὰ τὸ πρόσβορρον τοῦ Ταύρου πλευρὸν Καππάδοκες; αὕτη γὰρ ἡ χώρα τὸ πλέον πεδίων εὐπορεῖ καὶ χώρας ἀγαθῆς.

1–3 G T U V **3–10** G U

2 ἡ ... ἐστι: Konon Paraliotes, Κοσμογραφία VI 7

1 Κιλικίας: Πισιδίας V || ἔξω: *del.* G **3** ἢ σπανίως: ἢ σπανιαίτερον V U; *del.* T **7** καὶ[2]: *del.* G

Strabons Geographika XIV 5,1

Von Kilikien jenseits des Taurosgebirges wird ein Teil das Rauhe und ein Teil das Ebene genannt.[1] Das Rauhe ist der Teil, dessen Küste schmal ist und nirgends oder nur selten eine flache Stelle hat, sowie das Gebiet, über dem der Tauros liegt, kaum bewohnt bis ganz zu seinen nördlichen Flanken bei Isaura und den Homonadeern bis Pisidien.[2] Dasselbe Land wird auch Tracheiotis genannt und seine Einwohner Tracheioten. 2R 4R 6R 8R

Das Ebene ist der Teil, der von Soloi und Tarsos bis Issos reicht, sowie das Gebiet, über dem auf der nördlichen Flanke des Tauros die Kappadokier wohnen. Denn dieses Land ist zum größten Teil reich an Ebenen und guten Feldern. 10R 12R

[1]Dies ist eine Anmerkung zum Text. Sie ist nicht sehr lang.

[2]Dies ist noch eine Anmerkung zum Text. Sie ist deutlich länger und bedeutend gelehrsamer. Das allerdings hilft ihr auch nicht weiter.

Wie man einen eigenen TEXLive-Mirror aufsetzt

Uwe Ziegenhagen

In diesem Artikel möchte ich zeigen, wie man ein Netzwerkspeichersystem – auch »NAS« genannt – dazu benutzen kann, einen eigenen Spiegelserver von TEXLive für den Gebrauch im heimischen Netzwerk aufzusetzen.

Einführung

Als sehr »IT-affiner« Mensch geht es mir wie sicherlich vielen anderen Mitgliedern von DANTE e.V. auch: Ich habe zuhause (weit) mehr als einen Rechner. Und natürlich soll jeder dieser Rechner mit einem aktuellen TEXLive ausgestattet sein. Selbstverständlich ließe sich jeder Rechner über einen der öffentlichen CTAN-Server aktuell halten, ein eigener Server innerhalb des lokalen Netzes bietet aber einige Vorteile: Erstens verringert man dadurch sowohl die Last auf den CTAN-Servern, zweitens ist bei den meisten TEX-Nutzern das interne Netzwerk noch immer deutlich schneller sein als die externe Datenleitung. Als technische Basis für den Betrieb nutze ich ein NAS (»Networked Attached Storage«) der Firma Synology, das sowieso rund um die Uhr angeschaltet ist.

Kleiner Exkurs: Netzwerkfestplatten

NAS-Systeme sind grundsätzlich kleine eigenständige Rechner, die an das lokale Netzwerk angeschlossen sind und üblicherweise ausschließlich per Weboberfläche administriert werden, was je nach Anbieter mehr oder weniger komfortabel ist. Sie verfügen meist über mehr als eine Festplatte, die über ein RAID zusammengeschlossen sind und Schutz gegen Datenausfall bieten sollen. Ein Einwurf an dieser Stelle sei noch erlaubt: Ein RAID ersetzt kein Backup! Im Falle eines defekten RAID-Controllers sind die Daten auf allen Platten futsch, auch bei Diebstahl oder Wasser-/Feuerschaden nutzt die Redundanz durch das RAID nur wenig.

Zusammen mit der Firma QNAP ist Synology einer der Premiumhersteller, deren qualitativ sehr gute Geräte neben der eigentlichen Funktion eines Dateiservers noch viele andere Funktionen übernehmen können. Dem interessierten Leser sei die Herstellerwebseite `www.synology.de` empfohlen.

Schritt 1: Aktivierung der »Webstation«

Um später dem TEXLive-Manager eine URL anbieten zu können, von der die Pakete geladen werden können, installieren und aktivieren wir im ersten Schritt die

sogenannte »Webstation«, die nichts anderes ist als ein Apache-Webserver mit Unterstützung für PHP.

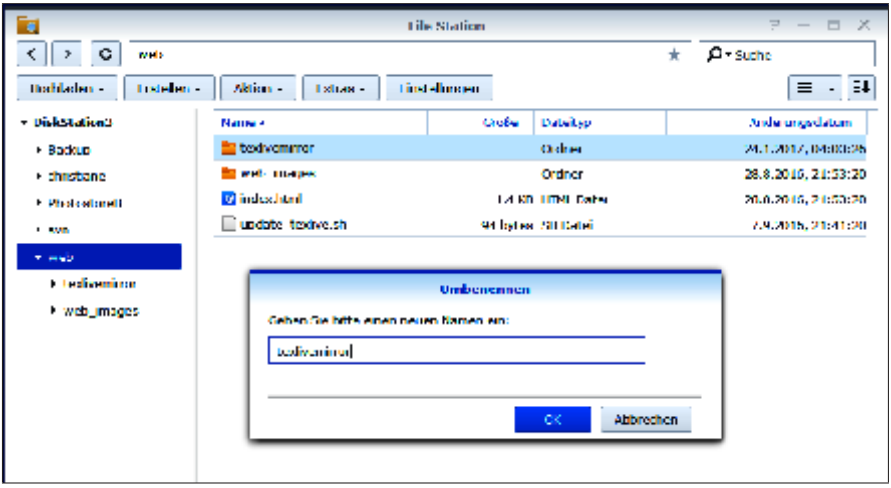


Abb. 1: Anlage des Unterordners `texlivemirror` im `web`-Ordner

Das Installationspaket findet man im Paket-Zentrum, nach der Installation gibt es einen neuen gemeinsamen Ordner `web`, der alle Webserver-Inhalte aufnimmt. Hier erstellen wir über die Weboberfläche einen neuen Ordner »texlivemirror«, in dem die Daten später abgelegt werden (Abbildung 1).

Schritt 2: Erstellung des `rsync`-Skripts

Für den Abgleich zwischen entferntem und lokalem Datenstand nutze ich `rsync`, ein seit langer Zeit bewährtes Programm zur Synchronisation von Dateien und Verzeichnissen. Anhand des Vergleichs von der Größe und des Zeitstempels gleicht `rsync` nur die Dateien ab, die sich seit der letzten Synchronisation geändert haben. Um das Skript für `rsync` zu erstellen, wird in unserem »Web«-Stammordner eine Datei `update_texlive.sh` angelegt, die mit dem Inhalt aus Listing 1 befüllt wird. Die einzelnen Parameter des `rsync`-Aufrufs sind dabei:

- `a` Fasst diverse Optionen für Unterverzeichnisse, symbolische Links und Rechte zusammen (siehe beispielsweise `https://wiki.ubuntuusers.de/rsync/`).
- `v` Zeigt während der Synchronisation die ausgeführten Schritte an.
- `rsync://...` Die Quelle von der abgeglichen wird.
- `/volume1/...` Das abzugleichende lokale Zielverzeichnis.

Da ich persönlich keine Möglichkeit gefunden habe, diese Datei anschließend über die Web-Oberfläche des NAS zu einem ausführbaren Skript zu machen, loggen wir uns mit dem `admin`-Account des NAS darauf ein (der `ssh` Dienst – den man in der Systemsteuerung unter »Terminal & SNMP« aktiviert – muss dazu laufen) und wechseln in den entsprechenden Ordner (`cd /volume1/web/`). Ein anschließendes `chmod +x update_texlive.sh` setzt das entsprechende Flag `execute`.

Listing 1: `update_texlive.sh`, das `rsync` Skript für die Synchronisation

```
rsync -av rsync://rsync.dante.ctan.org/CTAN/systems/texlive/tlnet/ /volume1/web/
 ↪texlivemirror/
```

Jetzt können wir das Skript bereits aufrufen und den initialen Datenstand vom entfernten Server holen. Listing 2 zeigt die ersten Zeilen des Skript-Aufrufs.

Listing 2: Die ersten Output-Zeilen des ersten Skript-Aufrufs

```
admin@DiskStation3:/volume1/web$ ./update_texlive.sh
receiving incremental file list
./
archive/
archive/12many.doc.tar.xz
archive/12many.source.tar.xz
archive/12many.tar.xz
archive/2up.doc.tar.xz
archive/2up.tar.xz
archive/Asana-Math.doc.tar.xz
archive/Asana-Math.tar.xz
archive/ESIEEcv.doc.tar.xz
archive/ESIEEcv.source.tar.xz
archive/ESIEEcv.tar.xz
archive/FAQ-en.doc.tar.xz
archive/FAQ-en.tar.xz
archive/GS1.doc.tar.xz
```

Automatischer Aufruf

Damit sich unser TEX-Spiegelserver die jeweils neuesten Inhalte zeitgesteuert holen soll, müssen wir nun dafür sorgen, dass das Skript regelmäßig aufgerufen wird. Was auf einer normalen Unix/Linux Maschine der `cronjob` ist, ist auf dem Synology NAS der Aufgabenplaner in der Systemsteuerung, siehe Abbildung 2. Hier erstellen wir über die drei Tabs (siehe Abbildungen 3 bis 4) einen neuen Task, der im Beispiel täglich um 01:00 Uhr nachts das Update-Skript aufruft.

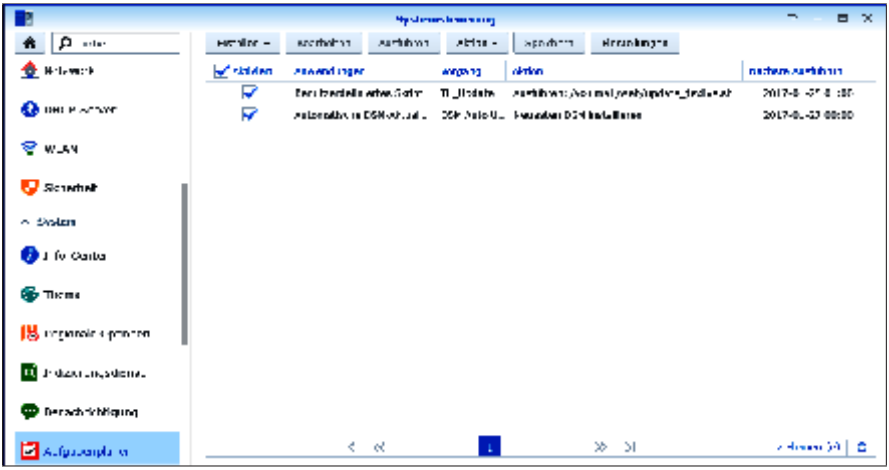


Abb. 2: Der Aufgabenplaner in der Systemsteuerung.

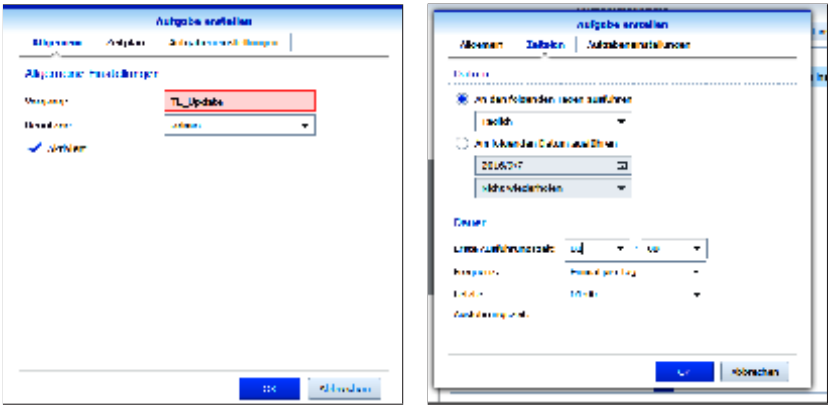


Abb. 3: Die Tabs »Allgemein« und »Aufgabe erstellen« der neuen Aufgabe.

Installation und Update mit dem eigenen Server

Bei einer Neu-Installation von TEXLive müssen wir jetzt nicht mehr das Installationsskript von `http://tug.org/texlive` holen, sondern könnten es über den Webbrowser von unserem NAS unter `http://‹IP-Adresse_des_NAS›/texlivemirror/` herunterla-

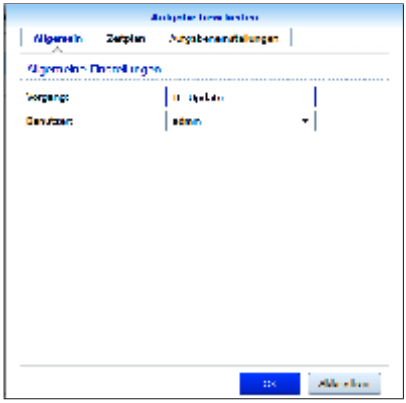


Abb. 4: Der Tab »Aufgabeneinstellungen« der neuen Aufgabe.

den. Jedoch fehlt noch ein Schritt, denn in der Standardkonfiguration listet die Webstation das Verzeichnis nicht auf, sondern liefert den Fehler 403 zurück. Es fehlt noch eine Datei `.htaccess` im Verzeichnis `texlivemirror`. Auch wenn ich persönlich Emacs bevorzuge: An dieser Stelle tut es – mangels installiertem Emacs – auch `vi`. Wir loggen uns wieder per SSH ein, wechseln nach `/volume1/web/texlivemirror` und geben den Befehl `vi .htaccess` ein. Im `vi` nutzen wir die Taste »i«, um in den Einfügemodus zu kommen und geben dann `Options +Indexes` ein. Mit `‹Escape›:wq` speichern wir und beenden `vi`.[1] Anschließend starten wir das Installations-Skript mittels (unter Windows)

```
install-tl-windows.bat --location http://‹IP-Adresse_des_NAS›/texlivemirror/
```

beziehungsweise (unter Linux/Unix)

```
install-tl --location http://‹IP-Adresse_des_NAS›/texlivemirror/
```

um TEXLive dem neuen Spiegelserver bekannt zu machen. Nach der Installation von TEXLive setzt man dann nur noch das Standard-Repository mittels

```
tlmgr option repository http://‹IP-Adresse_des_NAS›/texlivemirror
```

Fazit

Der Artikel hat am Beispiel eines Synology NAS gezeigt, wie man mit vergleichsweise wenig Aufwand einen eigenen Spiegelserver aufsetzen kann, um das lokale Netzwerk schnell und bequem mit Updates von TEXLive zu versorgen. Das gezeigte Vorgehen lässt sich leicht auf andere Umgebungen übertragen, auch ein Raspberry Pi könnte die Aufgabe leicht übernehmen.

[1] Mehr als die drei oder vier Tastenkombinationen werde ich mir persönlich nie merken können …

Strukturbäume mit Ti*k*Z

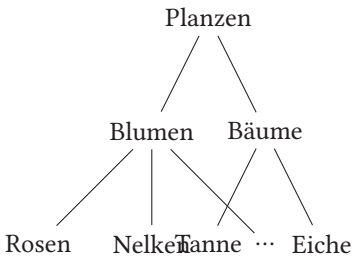
Christine Römer

Einordnung

Dieser kleine Beitrag versteht sich als Ergänzung zu »Konstituentenstrukturen einfach und schön mit forest«, wo das auf Ti*k*Z aufbauende Paket forest vorgestellt wurde. [4] Immer mal wieder gibt es jedoch Fragen zur Erstellung von »Bäumen« direkt mit Ti*k*Z, obwohl es doch eine sehr umfangreiche Dokumentation zum Paket gibt. Anhand von Beispielen, die bis auf die ersten aus dem Netz stammen, soll in das Erstellen von Baumstrukturen mit Ti*k*Z eingeführt werden. Dabei zeigt sich, wenn man den Bau der Bäume anschaut und durchdenkt, dass es gar nicht so schwer ist, wie es scheint, wenn man in der Dokumentation (PGF-Manual) von über 1 000 Seiten Umfang [5] etwas ratlos sucht. Man muss für das Erstellen auch nicht in alle Feinheiten von PGF/Ti*k*Z eindringen.

Die Astlängen und Astspreizungen festlegen

Wenn man eine kleine grafische Übersicht, wie die folgende, mit Ti*k*Z konstruiert, fällt auf, dass nicht wie bei dem Paket forest die Äste und Knotenabstände automatisch angepasst werden. Wenn keine Anpassung erfolgt, kann es zu Überlappungen kommen (wie in Beispiel 1).

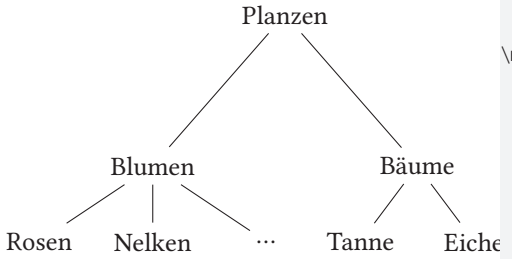



```
\begin{tikzpicture}
\node {Planzen}
  child {node {Blumen}
    child {node {Rosen}}
    child {node {Nelken}}
    child {node {\ldots}}
  }
  child {node {Bäume}
    child {node {Tanne}}
    child {node {Eiche}}
  };
\end{tikzpicture}
```

Beispiel 1: Ohne Abstandsregulierung

Wie in Beispiel 2 zu sehen ist, können die Werte für jede Gliederungsebene (= level) individuell festgelegt werden:

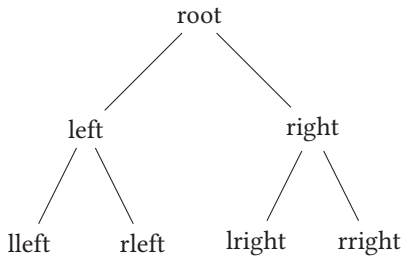
`sibling distance=` legt den horizontalen Abstand zwischen den Knoten fest,
`level distance=` die Höhe der Äste.




```
\begin{tikzpicture}
  [level 1/.style={sibling distance=35mm,
   level distance=20mm},
   level 2/.style={sibling distance=15mm,
   level distance=10mm}]
\node {Planzen}
  child {node {Blumen}
    child {node {Rosen}}
    child {node {Nelken}}
    child {node {\ldots}}
  }
  child {node {Bäume}
    child {node {Tanne}}
    child {node {Eiche}}
  };
\end{tikzpicture}
```

Beispiel 2: Regulierung des horizontalen Abstands

Mit dem Laden der Ti*k*Z-Bibliothek `trees` durch `\usetikzlibrary{trees}` in der Präambel kann man die Astlängen global für alle Knoten festlegen, wie das Beispiel 3 zeigt.



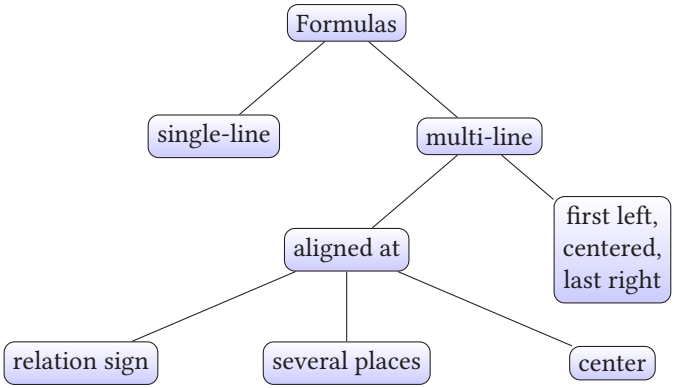

```
\begin{tikzpicture}
 [level distance=1.5cm,
  level 1/.style={sibling distance=3cm},
  level 2/.style={sibling distance=1.5cm}]
  \node {root}
    child {node {left}
      child {node {lleft}}
      child {node {rleft}}
    }
    child {node {right}
    child {node {lright}}
      child {node {rright}}
    };
\end{tikzpicture}
```

Beispiel 3: Globale Ebenenabstandsregulierung

Knoten individuell gestalten

Das Beispiel 4 zeigt auf, wie man Knoten u. a. mittels `every node/.style =...` individuell gestalten und mit `color=...` einfärben kann. [3]

```
\begin{tikzpicture}[sibling distance=10em,
  every node/.style = {shape=rectangle, rounded corners, draw, align=center,
  top color=white, bottom color=blue!20}]]
  \node {Formulas}
    child { node {single-line} }
    child { node {multi-line}
      child { node {aligned at}
        child { node {relation sign} }
        child { node {several places} }
        child { node {center} } }
      child { node {first left,\\centered,\\last right} } };
\end{tikzpicture}
```



Beispiel 4: »Ausgeschmückter Baum«

Äste vertikal anordnen

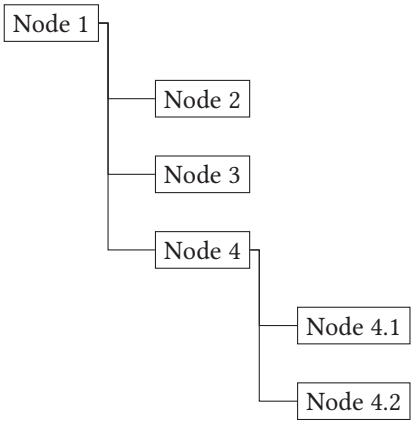
Wie in [5, Kap. 72] ausgeführt, kann man Knoten auch individuell (nicht als Standard ausgewiesen) positionieren. Dies geschieht mit der Option `/tikz/grow`:

`/tikz/grow via three points=one child at (x) and two children at (y) and (z)`

»Wenn ein übergeordneter Knoten nur ein Kind hat, wird es bei x platziert, wenn der übergeordnete Knoten zwei Kinder hat, werden diese bei y und z platziert.« (S. 751). Wie dies geschieht, wird aus dem folgenden Beispiel ([6]) ersichtlich. Es

ist für dieses Beispiel nötig, `\usetikzlibrary{arrows,positioning,scopes,trees}` zu laden.

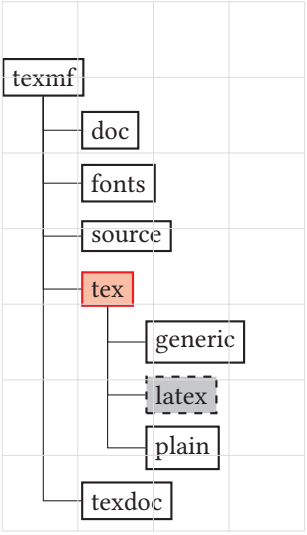
```
\begin{tikzpicture}[grow via three points={%
  one child at (2,-1) and two children at (2, -1) and (2, -2)},
  edge from parent fork right ]
   \node [draw, anchor=south west] {Node 1}
         child { node [draw] {Node 2} }
         child { node [draw] {Node 3} }
         child { node [draw] {Node 4}
            child { node [draw, edge from parent fork down] {Node 4.1} }
            child { node [draw] {Node 4.2} }
         };
\end{tikzpicture}
```



Beispiel 5: Horizontal orientierte »Bäume«

Das nächste Beispiel 6 soll das Augenmerk auf die Lokalisierung der Knoten lenken, die auch mit »geografischen« Orientierungen (`.south`, `.west`) vorgenommen wird. [1] Bei der Positionierung der Knoten kann das vorläufige Einfügen von Hilfslinien nützlich sein.

```
\tikzstyle{every node}=[draw=black,thick,anchor=west]
\tikzstyle{selected}=[draw=red,fill=red!30]
\tikzstyle{optional}=[dashed,fill=gray!50]
\begin{tikzpicture}[%
  grow via three points={one child at (0.5,-0.7) and
  two children at (0.5,-0.7) and (0.5,-1.4)},
  edge from parent path={(\tikzparentnode.south) |-
  (\tikzchildnode.west)}]
  \node {texmf}
    child { node {doc}}
    child { node {fonts}}
    child { node {source}}
    child { node [selected] {tex}
      child { node {generic}}
      child { node [optional] {latex}}
      child { node {plain}}
    }
    child [missing] {}
    child [missing] {}
    child [missing] {}
    child { node {texdoc}};
\draw[help lines,step=1.0,gray!30, very thin] (0,-6) grid (4,1) ;
\end{tikzpicture}
```

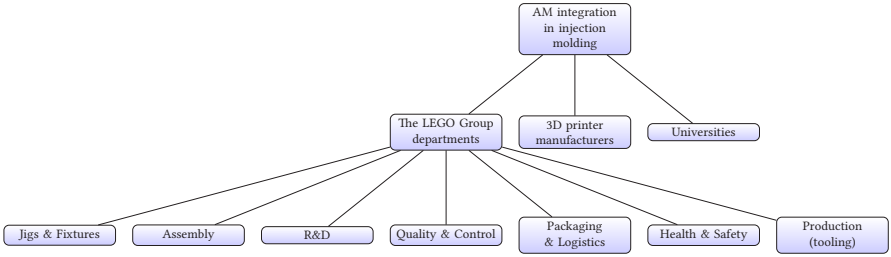


Beispiel 6: Horizontal orientierte »Bäume«

Vielzahl an Ästen

Wenn man eine größere Zahl von Ästen einfügen möchte, ist es hilfreich das Paket `adjustbox` zu verwenden, das auf einfache Weise eine zu große Grafik skalieren kann. [2] Diese wäre ansonsten größer als die vorgegebene Zeilenbreite.

```
\begin{adjustbox}{width=\textwidth}
\begin{tikzpicture}[sibling distance=10em, level distance=8em,
  every node/.style = {shape=rectangle, rounded corners, draw, align=center,
    text width=8em, top color=white, bottom color=blue!20}]]
  \node {AM integration \\ in injection molding}
    child { node {The LEGO Group departments}
        child { node {Jigs \& Fixtures}}
        child { node {Assembly}}
        child { node {R\&D}}
        child { node {Quality \& Control}}
        child { node {Packaging \& Logistics}}
        child { node {Health \& Safety} }
        child { node{Production (tooling)}}
        }
    child { node {3D printer manufacturers} }
    child { node {Universities}};
\end{tikzpicture}
\end{adjustbox}
```



Beispiel 7: Vielverzweigungen

Literatur und Software

[1] Frantisek Burian: Filesystem tree, 2011, `http://www.texample.net/tikz/examples/filesystem-tree/`.

[2] John Kormylo: Align tree, 2016, `http://tex.stackexchange.com/questions/344198/align-tikz-tree`.

[3] Stefan Kottwitz: A simple tree, 2015, http://www.texample.net/tikz/examples/tree/.

[4] Christine Römer: »Konstituentenstrukturen einfach und schön mit forest«, Die TEXnische Komödie, 2 (2016), 57–62.

[5] Till Tantau: The TikZ and PGF Packages – Manual for version 3.0.1a, 2015, http://sourceforge.net/projects/pgf.

[6] usr345: Vertical alignment of subnodes in a tree, 2011, http://latex-community.org/forum/viewtopic.php?t=12944.

Foto: Christine Römer (»MUSEO CASA DI DANTE« in Florenz)

Von fremden Bühnen

Im Netz gefunden

Herbert Voß

In den verschiedenen Mailinglisten, Webforen, Newsgroups u. a. findet man immer wieder hilfreiche Angaben zur Arbeit mit und um das Thema Textsatz mit TEX, LATEX, ConTEXt usw.

Wertebereich von \escapechar, \newlinechar, \endlinechar bei den verschiedenen TEX-Engines[1]

Meine Tests mit XETEX und LuaTEX haben folgendes ergeben:

a) \escapechar
   - XETEX und LuaTEX:
     Wird ein Wert kleiner als 0 oder größer als 1114111 angegeben, gibt \the\escapechar diesen Wert normal aus und beim Ausgeben der Namen von Kontrollsequenzen wird kein Zeichen vor den Namen gestellt.
     Wird ein Wert im Bereich von 0 bis 1114111 angegeben, gibt \the\escapechar diesen Wert normal aus, und beim Ausgeben der Namen von Kontrollsequenzen wird dasjenige Zeichen vor den Namen gestellt, dessen Character-Code dem als \escapechar angegebenen Wert entspricht.

b) \endlinechar
   - XETEX:
     Wird ein Wert kleiner als 0 oder größer als 255 angegeben, gibt \the\endlinechar diesen Wert normal aus und es wird kein Zeichen an eingelesene Zeilenenden angefügt.
     Wird ein Wert von 0 bis 255 angegeben, gibt \the\endlinechar diesen Wert normal aus und es wird dasjenige Zeichen an eingelesene Zeilenenden angefügt, dessen Character-Code dem als \endlinechar angegebenen Wert entspricht.

[1] Ulrich Diez am 13.12.2016 in de.comp.text.tex

- LuaTEX:
  Wird ein Wert kleiner als 0 angegeben, gibt `\the\endlinechar` diesen Wert normal aus und es wird kein Zeichen an eingelesene Zeilenenden angefügt.
  Wird ein Wert von 0 bis 127 angegeben, gibt `\the\endlinechar` diesen Wert normal aus und es wird dasjenige Zeichen an eingelesene Zeilenenden angefügt, dessen Character-Code dem als `\endlinechar` angegebenen Wert entspricht.
  Wird ein Wert größer als 127 angegeben, bekommt man die Fehlermeldung »Invalid `\endlinechar`« und das Assignment wird ignoriert.

c) `\newlinechar`

- XETEX:
  Wird ein Wert kleiner als 0 oder größer als 1114111 angegeben, gibt `\the\newlinechar` diesen Wert normal aus. Da alle von XETEX einlesbaren Zeichen positive Zahlen <= 1114111 als Character-Codes haben, bedeutet dies nur, dass derzeit kein einlesbares Zeichen als »Zeilenumbruchmarkierer« beim Schreiben von Text in eine externe Datei oder auf den Bildschirm fungiert.
  Wird ein Wert von 0 bis 1114111 angegeben, kann dasjenige Zeichen, dessen Character-Code dem angegebenen Wert entspricht, beim Schreiben von externen Dateien oder beim Schreiben auf den Bildschirm als Markierung für Zeilenumbrüche verwenden.
- LuaTEX:
  Wird ein Wert kleiner als 0 angegeben, gibt `\the\newlinechar` diesen Wert normal aus. Da alle von XETEX einlesbaren Zeichen positive Zahlen als Character-Codes haben, bedeutet dies nur, dass derzeit kein einlesbares Zeichen als »Zeilenumbruchmarkierer« beim Schreiben von Text in eine externe Datei oder auf den Bildschirm fungiert.
  Wird ein Wert von 0 bis 127 angegeben, kann man dasjenige Zeichen, dessen Character-Code dem angegebenen Wert entspricht, beim Schreiben von externen Dateien oder beim Schreiben auf den Bildschirm als Markierung für Zeilenumbrüche verwenden.
  Wird ein Wert größer als 127 angegeben, wird das `\newlinechar`-Assignment ignoriert und man bekommt die Fehlermeldung »Invalid `\newlinechar`«.

Erzeugen einer Referenz auf eine Gleichung innerhalb von `\lstinputlisting`[2]

Aus einem Listing heraus soll eine Referenz auf eine Gleichung im normalen Text gesetzt werden. Im Python-Quellcode steht die Referenz zwischen zwei Kommentarzeichen %:

```
% Equation \ref{eq:my equation} %
```

Das Kommentarzeichen wird jetzt der Option `escapeinside` mitgeteilt, sodass `\ref`, obwohl mit `\lstinputlisting` eingelesen, ausgewertet wird und in der Ausgabe des Quellcodes »Equation 1« erscheint.

```
\begin{filecontents*}{\jobname.py}
#code.py
import math
def y(x):
    return math.sin(x)*math.cos(x) # % Equation \ref{eq:my equation} %
print y(math.pi/4)
\end{filecontents*}

\begin{align}\label{eq:my equation}
  y = \sin(x) \cos(x)
\end{align}

\lstinputlisting[caption={Python-Listing mit Referenz},language=Python,
   frame=single,escapeinside=\%\%]{\jobname.py}
```

$$y = \sin(x)\cos(x) \tag{1}$$

Listing 1: Python-Listing mit Referenz

```
#code.py
import math
def y(x):
    return math.sin(x)*math.cos(x) #  Equation 1
print y(math.pi/4)
```

[2] Paul Gaborit am 26.1.2017 in `http://tex.stackexchange.com/questions/350470/`.

Neue Pakete auf CTAN

Jürgen Fenn

Der Beitrag stellt neue Pakete auf CTAN seit der letzten Ausgabe vor. Bloße Updates können auf der moderierten CTAN-ann-Mailingliste verfolgt werden, die auch auf Twitter als @ctanannounce verfügbar ist.

undergradmath von Jim Hefferon ist ein LaTeX-Spickzettel zum mathematischen Formelsatz, der sich an (amerikanische) Studienanfänger richtet.
CTAN:info/undergradmath

platexcheat von Takuto Asakura ist eine japanische Übersetzung von Winston Changs LaTeX-Spickzettel mit Anpassungen zur Verwendung des Programms platex.
CTAN:info/platexcheat

halloweenmath von Gustavo Mezzetti stellt halloween-artige »Operatoren« für den Mathematiksatz bereit.
CTAN:macros/latex/contrib/halloweenmath

arimo von Bob Tennent enthält die gleichnamige Schriftart samt der dazu gehörigen LaTeX-Unterstützung. Die Schrift ist hinsichtlich ihrer Metriken kompatibel mit der Arial.
CTAN:fonts/arimo

mpostinl von Niklas Beisert hilft beim Einbetten einer METAPOST-Grafik in ein LaTeX-Dokument.
CTAN:macros/latex/contrib/mpostinl

tinos von Bob Tennent enthält die Serifenschrift Tinos von Steve Matteson samt der dazu gehörigen LaTeX-Unterstützung. Die Schrift ist hinsichtlich ihrer Metriken kompatibel mit der Times New Roman.
CTAN:fonts/tinos

simple-thesis-dissertation von Zach Scrivena enthält eine einfache Vorlage zum Schreiben einer wissenschaftlichen Abschlussarbeit mit XeLaTeX.
CTAN:macros/xetex/latex/simple-thesis-dissertation

simple-resume-cv von Zach Scrivena ist eine Dokumentenklasse zum Setzen eines Lebenslaufs für berufliche Bewerbungsunterlagen mit XeLaTeX.
CTAN:macros/xetex/latex/simple-resume-cv

missaali von Tommi Syrjänen enthält die gleichnamige freie Schriftart im Format OpenType samt der dazu gehörigen Unterstützung für XeLaTeX. Die Missaali ist einer gebrochenen Schrift nachempfunden, die der deutsche Drucker und Typograf Bartholomäus Ghotan im ausgehenden 15. Jahrhundert zum Setzen von Messbüchern verwendet hatte.
CTAN:fonts/missaali

yaletter (kurz für: yet another letter) von Donald P. Goodman ist eine neue Dokumentenklasse zum Setzen von Briefen, Umschlägen und Adressetiketten nach den Gepflogenheiten in den USA. Die Anschriften können in einer Datenbank bereitgehalten werden.
CTAN:macros/latex/contrib/yaletter

pst-shell von Manuel Luque und Herbert Voß dient zum Zeichnen von dreidimensionalen Schnecken- und Muschelformen mit Hilfe von PSTricks.
CTAN:graphics/pstricks/contrib/pst-shell

css-colors von Engelbert Buxbaum erweitert das Paket `color` von D. P. Carlisle um die Bezeichnungen für die 143 websicheren Farben. Sie sind z. B. hilfreich, wenn es darauf ankommt, bestimmte Farben für die Gestaltung von Präsentationen auszuwählen.
CTAN:macros/latex/contrib/css-colors

scsnowman von Hironobu Yamashita dient dazu, Schneemänner mit Hilfe von Ti*k*Z zu zeichnen. Die dabei möglichen Varianten sind an die ursprüngliche Implementierung in Unicode angelehnt, gehen aber darüber hinaus.
CTAN:graphics/pgf/contrib/scsnowman

stanli von Jürgen Hackl ist eine Ti*k*Z-Library für zwei- und dreidimensionale Darstellungen zur Baustatik (STructural Analysis LIbrary). Die damit zu erstellenden Zeichnungen sind vor allem für die Lehre wichtig.
CTAN:graphics/pgf/contrib/stanli

conv-xkv von Donald P. Story ermöglicht es, alternative Schreibweisen für Schlüsselwertparameter (key–value) zu definieren, die an das Paket `xkeyval` durchgereicht werden.
CTAN:macros/latex/contrib/conv-xkv

mendex-doc von der Japanese TEX Development Community ist die japanische Anleitung zu dem Indexprozessor Mendex, der in TEXLive 2016 enthalten ist.
CTAN:info/mendex-doc

arphic-ttf von Hironobu Yamashita enthält die TrueType-Version der chinesischen Schriftart Arphic zur Verwendung mit XeLATEX und LuaLATEX.
CTAN:fonts/arphic-ttf

luahyphenrules von Javier Bezos dient dazu, die Trennmuster für eine Sprache mit LuaTEX zu laden, wenn das Paket `babel` nicht verwendet wird.
CTAN:macros/luatex/latex/luahyphenrules

dtxdescribe von Brian Dunn erweitert die Möglichkeiten zum Beschreiben von Makros und Umgebungen in dem Standardpaket `doc` um eine Reihe von Parametern, Optionen usw.
CTAN:macros/latex/contrib/dtxdescribe

keyfloat von Brian Dunn stellt ein umfangreiches Key-Value-Interface für die Positionierung und den weiteren Umgang mit Gleitobjekten in einem Dokument bereit.
CTAN:macros/latex/contrib/keyfloat

padauk von Bob Tennent enthält jeweils zwei Schnitte der Schriftarten Padauk und Padauk Book im Format TrueType zum Setzen von Texten in birmanischen Sprachen.
CTAN:fonts/padauk

footmisx von Bastien Roucaries beruht auf `footmisc` von Robin Fairbairns. Es bietet alle Features des ursprünglichen Pakets, arbeitet darüber hinaus aber auch mit dem Paket `hyperref` zusammen. Eine Version für LATEX3 soll folgen.
CTAN:macros/latex/contrib/footmisx

awesomebox von Étienne Deparis erzeugt Info- und Warnkästen und stützt sich dabei auf `fontawesome.sty` von Honza Ustohal.Dazu werden LuaLaTeX oder XeLaTeX sowie die Schriftart Font Awesome von `http://fontawesome.io/` benötigt.
`CTAN:graphics/awesomebox`

karnaugh-map von Mattias Jacobsson zeichnet Karnaugh-Veitch-Diagramme zur Darstellung Boolescher Funktionen mit bis zu sechs Variablen.
`CTAN:graphics/pgf/contrib/karnaugh-map`

gofonts von Bob Tennent enthält eine serifenlose und eine Festbreitenvariante der Fonts, die Bigelow & Holmes für das Go-Projekt geschaffen hatten, einschließlich der LaTeX-Unterstützung.
`CTAN:fonts/gofonts`

baskervillef von Michael Sharpe ist eine modifizierte und erweiterte Version der Schriftart Baskerville in der Version von Joseph Fry, die von der Libre Baskerville abgeleitet wurde, in den Formaten OpenType und PostScript, samt der LaTeX-Unterstützung.
`CTAN:fonts/baskervillef`

forest-quickstart von Guido Vanden Wyngaerd ist eine knapp gehaltene zwölfseitige Anleitung zu dem Paket `forest` von Sašo Živanović, mit dem Sprachwissenschaftler Baumdiagramme zeichnen können.
`CTAN:info/forest-quickstart`

fribidixetex von Vafa Khalighi ist ein Präprozessor, mit dem man Dokumente für bidirektional zu setzende Texte zur weiteren Bearbeitung mit XeTeX vorbereiten kann.
`CTAN:support/fribidixetex`

unicode-bidi von Vafa Khalighi ist ein experimentelles Paket für den bidirektionalen Textsatz mit XeTeX, bei dem man darauf verzichten kann, die Satzrichtung (rechts/links) auszuzeichnen.
`CTAN:macros/xetex/latex/unicode-bidi`

tikzpeople von Nils Fleischhacker stellt stilisierte Zeichnungen von Personen bereit, die in Diagrammen und Grafiken verwendet werden können.
`CTAN:graphics/pgf/contrib/tikzpeople`

delimset von Niklas Beisert soll die Eingabe von Klammernpaaren im Mathematikmodus erleichtern.
`CTAN:macros/latex/contrib/delimset`

apxproof von Pierre Senellart platziert mathematische Beweise und ähnliches Material, das im Fließtext eingegeben worden ist, in den Anhang des Dokuments samt einer separaten Bibliografie für diesen Teil.
`CTAN:macros/latex/contrib/apxproof`

wtref von Takuto Asakura erweitert die Möglichkeiten für Querverweise in LaTeX um neue Befehle, die man für Namensräume und Zählerintervalle definieren kann.
`CTAN:macros/latex/contrib/wtref`

biblatex-gb7714-2015 von Hu Zhenzhen ist eine Implementierung des chinesischen Zitierstils GBT7714-2015 für BibLaTeX.
`CTAN:macros/latex/contrib/biblatex-contrib/biblatex-gb7714-2015`

banglatex von Adib Hasan dient zum Setzen von Texten in bengalischer Sprache.
CTAN:language/banglatex

latex2nemeth von Antonis Tsolomitis und Andreas Papasalouros kann LaTeX-Dokumente, einschließlich mathematischer Formeln, ohne einen externen Konverter direkt in Nemeth Braille umsetzen. Zum Prägen der Braille-Ausgabe wird LibreOffice mit `odt2braille` benötigt.
CTAN:support/latex2nemeth

testidx von Nicola Talbot erzeugt einen englischen Blindtext, mit dem man die Index-Erzeugung testen kann. Geprüft werden auch lateinische diakritische Zeichen, sowohl in der traditionellen LaTeX-Eingabe mit Akzent-Befehlen als auch direkt im Standard UTF-8.
CTAN:macros/latex/contrib/testidx

latex-papersize von Silas Brown berechnet LaTeX-Einstellungen für jede Schriftart und für jedes Papierformat. Das Paket hieß früher `latexpaper`.
CTAN:support/latex-papersize

xcolor-material von Jerick Órdenes erweitert das Paket `xcolor` um die Google Material Color Palette, die Google zur Gestaltung seiner Webseiten entwickelt hat.
CTAN:macros/latex/contrib/xcolor-material

platex-tools von Hironobu Yamashita enthält Erweiterungen zu dem Paket `latex-tools` für Texte in japanischer Sprache, die mit pLaTeX/upLaTeX gesetzt werden.
CTAN:language/japanese/platex-tools

uppunctlm von Yuwsuke Kieda sorgt bei der Verwendung der Latin-Modern-Fonts dafür, dass die Satzzeichen Punkt, Komma, Bindestrich, die eckigen und die runden Klammern sowie die arabischen Ziffern Null bis Neun stets aufrecht (upright) gesetzt werden, auch wenn der umgebende Text kursiv ausgezeichnet ist.
CTAN:fonts/uppunctlm

biblatex-sbl von David Purton setzt den Zitierstil des Handbook of Style der Society of Biblical Literature für BibLaTeX um.
CTAN:macros/latex/contrib/biblatex-contrib/biblatex-sbl

Bücher

Edition dante – Neuauflage

Herbert Voß:

Präsentationen mit LaTeX; 2., überarbeitete und erweiterte Auflage Januar 2017,
DANTE e.V. und Lehmanns Media,
239 Seiten; ISBN 978-3-86541-836-4;
19,95 € (Ladenpreis) bzw. 17,– € 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 per PayPal. Die Kontonummer finden Sie am Ende dieses Heftes und Informationen zu PayPal auf `http://www.dante.de/index/Intern/Zahlung.html`.

Bitte beachten Sie für Bestellungen bei DANTE e.V. folgende Informationen zum Widerrufsrecht: Käufer 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. Der Besteller hat in jedem Fall die Rücksendekosten zu tragen. Bei Verschlechterung der Ware, die über die übliche Prüfung der Ware hinausgeht, hat der Besteller gegebenenfalls Wertersatz zu leisten.

Spielplan

2017

11. 3. – 12. 3. **Chemnitzer Linuxtage 2017**
Technische Universität Chemnitz
Reichenhainer Straße 90, 09126 Chemnitz
https://chemnitzer.linux-tage.de/2017/de/

22. 3. – 24. 3. **DANTE 2017**
und 56. Mitgliederversammlung von DANTE e.V.
Deutsches Elektronen-Synchotron (DESY)
Platanenallee 6, 15738 Zeuthen
http://www.dante.de/events/dante2017.html

3. 4. **16. Informatiktag NRW**
Bergische Universität Wuppertal und
Arbeitsgruppe Angewandte Informatik
(Fachgebiet Didaktik der Informatik)
http://informatiktag-nrw.de

29. 4. – 3. 5. **TUG 2017** zusammen mit
BachoTEX-Konferenz 2017
Bachotek, nahe Brodnica, Polen
http://www.tug.org/tug2017/
http://www.gust.org.pl/bachotex/

15. 5. – 16. 5. **PDF Days Europe 2017**
eine Veranstaltung der PDF Association
Berlin
https://www.pdfa.org/save-the-date-pdf-days-europe-2017/

11. 9. – 17. 9. **11th International ConTEXt Meeting**
35510 Butzbach – Alt-Maibach
http://meeting.contextgarden.net/2017/

2018

März **DANTE 2018**
und 58. Mitgliederversammlung von DANTE e.V.
Universität Passau

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://projekte.dante.de/Stammtische/WebHome.

Aachen
Torsten Bronger,
bronger@physik.rwth-aachen.de
Gaststätte Knossos, Templergraben 28, 52062 Aachen
Zweiter Donnerstag im Monat, 19.00 Uhr

Berlin
Michael-E. Voges, Tel.: (03362) 50 18 35,
mevoges@t-online.de
Mantee – Café Restaurant, Chausseestraße 131, 10115 Berlin
Zweiter Donnerstag im Monat, 19.00 Uhr

Bremen
Winfried Neugebauer, Tel.: 0176 60 85 43 05,
tex@wphn.de
Wechselnder Ort
Erster Donnerstag im Monat, 18.30 Uhr

Dresden
Daniel Borchmann, daniel@algebra20.de, http://tug-dd.dtnet.org
auf Anfrage

Erlangen
Walter Schmidt, Peter Seitz,
w.a.schmidt@gmx.net
Gaststätte »Deutsches Haus«, Luitpoldstraße 25, 91052 Erlangen
Dritter Dienstag im Monat, 19.00 Uhr

Frankfurt
Harald Vajkonny,
http://wiki.lug-frankfurt.de/TeXStammtisch
Restaurant »Zum Jordan«, Westerbachstr. 7, 60489 Frankfurt
Zweimonatlich, Vierter Donnerstag im Monat, 19.30 Uhr

Göttingen
Holger Nobach,
holger.nobach@nambis.de, http://goetex.nambis.de/
Restaurant Mazzoni Cucina Italiana,
Hermann-Rein-Straße 2, 37075 Göttingen
Dritter Donnerstag im Monat, 18.00 Uhr

Hamburg I
Lothar Fröhling,
lothar@thefroehlings.de
Letzter Dienstag im Monat an wechselnden Orten, 19.00 Uhr

Hamburg II
Günther Zander,
guenther.zander@lug-balista.de, http://www.lug-hamburg.de/kalender
Bürgerhaus in Barmbek, Lorichsstraße 28a, 22307 Hamburg
Zweiter Montag im Monat, 20.00 Uhr

Hannover
Mark Heisterkamp,
heisterkamp@rrzn.uni-hannover.de
Seminarraum RRZN, Schloßwender Straße 5, 30159 Hannover
Zweiter Donnerstag im Monat, 18.30 Uhr

Heidelberg
Martin Wilhelm Leidig, Tel.: 0170 418 33 29,
moss@moss.in-berlin.de
Anmeldeseite zur Mailingliste: http://tinyurl.com/stammtisch-HD
Wechselnder Ort
Letzter Freitag im Monat, ab 19.30 Uhr

Köln
Uwe Ziegenhagen
Dingfabrik, Erzbergerplatz 9, 50733 Köln
Zweiter Dienstag im Monat, 19.00 Uhr

München
Uwe Siart,
uwe.siart@tum.de, http://www.siart.de/typografie/stammtisch.xhtml
Erste Woche in geradzahligen Monaten an wechselnden Tagen, 20.00 Uhr

Stuttgart
Bernd Raichle,
bernd.raichle@gmx.de
»Trollinger-Stubn«, Rotebühlstr. 50, 70178 Stuttgart
Zweiter Dienstag im Monat, 19.30 Uhr

Trier
Martin Sievers,
ttt@schoenerpublizieren.de
Anmeldeseite zur Mailingliste: http://lists.schoenerpublizieren.de/cgi-bin/mailman/listinfo/ttt
Universität Trier
nach Vereinbarung

Wuppertal
Andreas Schrell, Tel.: (02193) 53 10 93,
as@schrell.de
Restaurant Croatia »Haus Johannisberg«, Südstraße 10, 42103 Wuppertal
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: info@dante.de

Konto: VR Bank Rhein-Neckar eG
IBAN DE67 6709 0000 0002 3100 07
SWIFT-BIC GENODE61MA2

Vorstand

Vorsitzender:	Martin Sievers	president@dante.de
stv. Vorsitzender:	Herbert Voß	vice-president@dante.de
Schatzmeisterin:	Doris Behrendt	treasurer@dante.de
Schriftführer:	Manfred Lotz	secretary@dante.de
Beisitzer:	Harald König	
	Volker RW Schaa	
	Dominik Wagenführ	
	Uwe Ziegenhagen	

Ehrenmitglieder

Peter Sandner	22.03.1990	Klaus Thull († 2012)	22.03.1990
Yannis Haralambous	05.09.1991	Barbara Beeton	27.02.1997
Luzia Dietsche	27.02.1997	Donald E. Knuth	27.02.1997
Eberhard Mattes	27.02.1997	Hermann Zapf († 2015)	19.02.1999
Joachim Lammarsch	12.04.2014	Rainer Schöpf	12.04.2014

Webserver und Mailingliste

DANTE:	http://www.dante.de/	(Rainer Schöpf, Joachim Schrod)
CTAN:	http://mirror.ctan.org/	(Gerd Neugebauer)
DANTE-EV:	https://lists.dante.de/mailman/listinfo/dante-ev	

FAQ

DTK: http://projekte.dante.de/DTK/WebHome
TEX: http://projekte.dante.de/DanteFAQ/WebHome

Autoren/Organisatoren

Jürgen Fenn [82]
Friedensallee 174/20
63263 Neu-Isenburg
juergen.fenn@gmx.de

Bogusław Jackowski [13]
Gdańsk, Polen
b_jackowski@gust.org.pl

Piotr Pianowski [13]
Trąbki Wielkie, Polen
p.pianowski@gust.org.pl

Philipp Pilhofer [53]
Bötzowstraße 45
10407 Berlin
philipp.pilhofer@kilikien.de

Henning Hraban Ramm [6]
texml@fiee.net

Christine Römer [72]
Iltisweg 39
07749 Jena
tex@christine-roemer.de

Martin Sievers [4]
siehe Seite 90

Piotr Strzelczyk [13]
Sopot, Polen
p.strzelczyk@gust.org.pl

Herbert Voß [3,79,86]
Wasgenstraße 21
14129 Berlin
herbert@dante.de

Uwe Ziegenhagen [67]
Lokomotivstraße 9
50733 Köln
ziegenhagen@gmail.com

Die TEXnische Komödie

29. Jahrgang Heft 1/2017 Februar 2017