Die TEXnische Komödie

dante
Deutschsprachige
Anwendervereinigung TEX e.V.

36. Jahrgang Heft 4/2024 November 2024

4/2024

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. Reproduktion oder Nutzung der erschienenen Beiträge durch konventionelle, elektronische oder beliebige andere Verfahren ist nicht gestattet. Alle Rechte zur weiteren Verwendung außerhalb von DANTE e.V. liegen bei den jeweiligen Autoren.

Beiträge sollten in Standard-LATEX-Quellcode unter Verwendung der Dokumentenklasse `dtk` erstellt und per E-Mail oder Datenträger (z. B. CD/DVD) an unten stehende 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 `https://projekte.dante.de/DTK/AutorInfo` von DANTE e.V.

Diese Ausgabe wurde mit `LuaHBTeX, Version 1.18.0 (TeX Live 2024)` erstellt. Als Standardschriften kamen Libertinus Serif, Libertinus Sans Serif, Anonymous Pro und Libertinus Math zum Einsatz.

Erscheinungsweise: vierteljährlich
Erscheinungsort: Heidelberg
Auflage: 2000
Herausgeber: DANTE, Deutschsprachige Anwendervereinigung TEX e.V.
Bergheimer Straße 147
69115 Heidelberg

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

Druck: Schleunungdruck GmbH
Eltertstraße 27, 97828 Marktheidenfeld

Redaktion: Luzia Dietsche (verantwortliche Redakteurin)
Mitarbeit: Gert-Ludwig Ingold, Rolf Niepraschk, Stefan Pinnow, Bernd Raichle, Christine Römer, HG Unckell, Herbert Voß

Redaktionsschluss für Heft 1/2025: 15. Januar 2025 ISSN 1434-5897

Editorial

Liebe Leserinnen und Leser,

ich freue mich, auch in dieser Ausgabe wieder eine schöne Mischung von Vereinsinterna und allgemeinen Themen präsentieren zu können.

Dem Aufruf zur nächsten Tagung folgen hoffentlich viele langjährige und neue Mitglieder. Es macht eben viel Spaß, sich in »real life« kennenzulernen und austauschen zu können. Vom Bericht zur letzten TUG-Tagung hat mich leider nur der erste Teil bis zum Redaktionsschluss erreicht, aber vielleicht kann er ja als Anreiz dienen, sich den einen oder anderen Vortrag im YouTube-Kanal der TEX Users Group anzusehen.

Auch diesmal freue ich mich über den Beitrag von Herrn Wehr über LATEX in der Schulphysik. Ob ich dem Schulfach wohl mehr hätte abgewinnen können, wenn ich es auf diese Weise kennengelernt hätte? Der Beitrag über variable Schriften wird mich auf jeden Fall aktuell dazu anregen, mehr mit Schriften zu spielen, wenn das nun so leicht geht. Hraban Ramm behauptet zwar jedes Mal, dass sich nicht viel in der ConTEXt-Welt getan hat, findet aber immer wieder Interessantes zu berichten. Frank Mittelbach und als Übersetzer Thomas Demmig lassen uns dankenswerterweise erneut an der Entwicklung von LATEX teilnehmen.

Und intern, also im Redaktionsteam, freue ich mich mit jeder Ausgabe darüber, wie reibungslos die Zusammenarbeit funktioniert. Der im Grußwort angesprochene Wechsel des Servers hat auch unsere Arbeit betroffen und verlief dank Erik Braun und Oliver Kopp reibungslos. Immer wieder bin ich fasziniert, wie genau die Redaktionsmitglieder die Texte lesen und Typos finden, die ich manchmal auch bei mehrmaligem Lesen nur schwer finde. Danke Team!

Damit wünsche ich auch diesmal viel Spaß beim Lesen und verbleibe mit TEXnischen Grüßen

Luzia Dietsche

PS: Bitte beachten – DANTE e.V. hat eine (mittlerweile nicht mehr ganz so) neue postalische Adresse (siehe Impressum).

Hinter der Bühne

Vereinsinternes

Grußwort

Liebe Mitglieder,

ich denke, man kann mit Fug und Recht sagen, dass 2024 das Jahr der Umzüge war und ist. Beginnend im letzten Jahr haben wir im März endlich den Auszug aus unserem langjährigen physischen Büro in Heidelberg abgeschlossen und residieren seitdem offiziell auf der anderen Straßenseite in einem virtuellen Büro. Die Post wird eingescannt, das Telefon vor Ort ist ebenfalls einer Onlinelösung gewichen. Für alles, was noch gebraucht wurde, nutzen wir eine Self-Storage-Lösung.

Ein weiterer komplexer Umzug stand vor einigen Wochen an. »Never change a running system« – diese »Weisheit«, die wie die Begriffe »Handy« oder »Public Viewing« zwar Englisch daherkommt, ihren Ursprung aber wohl eher in Deutschland hat, ist vielen von uns geläufig. Unser Verein hat sich lange daran orientiert, teilweise aus freien Stücken, teilweise unfreiwillig. Doch nun, am 30. September, war der Tag gekommen, an dem Dienste und Daten, die auf unserem Server »comedy« liefen, endgültig einen neuen Platz gefunden haben mussten.

Wir haben versucht, den Übergang gut vorzubereiten und zu gestalten, wobei die Hauptlast sicherlich bei Erik Braun lag. Unterstützung kam u. a. von Oliver Kopp und dem CTAN-Team, die durch den Umzug unmittelbar betroffen waren. Allen Beteiligten danke ich für ihr Engagement und ihre Geduld. Wir werden vermutlich noch eine Weile mit den Folgen des Umzugs zu tun haben. Es hakt noch an so mancher Stelle, dafür konnten wir andernorts überkommene Strukturen beenden.

Zwischen den beiden großen Umzugsereignissen lag die Frühjahrstagung, die Anfang April in Weimar stattfand. Wo einst Goethe und Schiller wandelten und das Bauhaus plante und ausbildete, tagten wir. Genauso spannend und interessant wie der Tagungsort waren auch die Vorträge und das abwechslungsreiche Begleitprogramm. An dieser Stelle nochmals ein herzliches Dankeschön an Thomas H. Meyer, der Maßstäbe gesetzt hat.

Für eine Herbsttagung haben die Ressourcen in diesem Jahr leider nicht gereicht. Denn auch aktuell beschäftigt den Vorstand – und ganz besonders unsere Schatzmeisterin Doris – ein Umzug. Diesmal geht es um die Mitgliederverwaltung und

Buchungssoftware. Leider mussten wir unsere Entscheidung für die Plattform Campai revidieren, nachdem sich immer mehr Defizite zeigten und wir ein ums andere Mal bei der Umsetzung von für uns wesentlichen Features vertröstet worden waren.

Wir sind nun doch wieder bei VEWA gelandet, einer »großen« Lösung, die uns helfen wird, Daten der Mitglieder sowie Buchungen effizient zu be- bzw. verarbeiten. Um einen »Arbeitszustand« zu erreichen, müssen allerdings einige Abstimmungsrunden gedreht werden, viel Papier bedruckt und gelesen sowie Dinge vorbereitet, getestet und entschieden werden. Im Laufe des ersten Quartals des neuen Jahres steht uns dann voraussichtlich eine VEWA-Instanz als Cloudlösung zur Verfügung. Bis dahin nutzen wir Campai für die Datenpflege weiter und buchen für das laufende Jahr in der alten lokalen VEWA-Instanz. Also hat es sich letztlich auch hier bewahrheitet: »Never change a running system.«

Zum Schluss noch ein Wort zur anderen »Weisheit«, die man oft im Sport findet: »Never change a winning team.« Hier können wir uns als DANTE e.V. aus meiner Sicht glücklich schätzen, dass es einige Personen gibt, die sich schon seit sehr vielen Jahren im Vorstand und den für uns wichtigen Gruppen wie der Mitgliederzeitung oder CTAN engagieren. Das ist nicht selbstverständlich und das aktuelle Jahr hat gezeigt, dass die reine Zahl von Vorstandsmitgliedern kein Garant für eine effiziente Aufteilung von Aufgaben ist.

Bei der Frühjahrstagung im April 2025 in Darmstadt (Einladung und »Call for Presentations« in dieser Ausgabe) wird der Vorstand neu gewählt. Es wird personelle Änderungen geben und ich lade Euch alle schon heute herzlich ein, aktiv daran mitzuwirken. Nur so können wir das, was in diesem Jahr neu aufgegleist und angestoßen wurde, mit voller Kraft zur Entfaltung bringen. Es braucht immer wieder neue Mitglieder, damit ein Verein nicht irgendwann seinen Sinn verliert. Doch genauso wichtig ist die regelmäßige Erneuerung des Vorstands, ohne den ein Verein – nicht nur rechtlich – nicht existieren kann. Es braucht Engagement, aber eben auch Verantwortung.

Jetzt im Herbst folgen die nächsten Umzüge, doch haben die zum Glück wenig mit DANTE e.V., dafür umso mehr mit Laternen und meinem Namenspatron zu tun. Danach folgt die so oft beschworene besinnliche Zeit. Möge es für uns alle eine solche werden. Ich wünsche Ihnen und Euch einen guten Jahresausklang und für das neue Jahr alles erdenklich Gute.

Und nun viel Spaß mit der Lektüre dieser Ausgabe der Mitgliederzeitung.

Herzlichst Ihr/Euer
Martin Sievers

Einladung zur Frühjahrstagung 2025 und 67. Mitgliederversammlung von DANTE e.V. an der Hochschule Darmstadt

Martin Sievers, Torsten-Karl Strempel

Liebe Mitglieder von DANTE e.V.,

wir laden Sie ganz herzlich zur Frühjahrstagung 2025 vom 3.–5. April 2025 in Darmstadt ein.

Der Zeitplan sieht wie folgt aus:

Mittwoch, 2. April,	ab 19 Uhr:	Vorabendtreff
Donnerstag, 3. April,	9 bis 17 Uhr:	Vorträge
	ab 19 Uhr:	Abendtreff
Freitag, 4. April,	9 bis 12 Uhr:	Vorträge
	ab 15 Uhr:	Besuch der ESA
	ab 19 Uhr:	Tagungsessen
Samstag, 5. April,	ab 9 Uhr:	67. Mitgliederversammlung
	anschließend:	Vorträge sowie Kulturprogramm

Die Adresse des Tagungsorts lautet:
h_da Hochschule Darmstadt
Gebäude C10
Schöfferstraße 3
64295 Darmstadt

Die Tagesordnung der 67. Mitgliederversammlung lautet:

1. Begrüßung und Tagesordnung
2. Bericht des Vorstands
3. Finanzbericht
4. Bericht der Rechnungsprüfer
5. Entlastung des Vorstands
6. Wahl eines Vorstands
7. Wahl von Rechnungsprüfern
8. Förderung von TEX-Projekten
   - Bericht über den aktuellen Stand
   - Antrag des Vorstands zur Bereitstellung neuer Projektmittel
9. Verschiedenes

Anträge zur Ergänzung oder Änderung der Tagesordnung sind gemäß § 12 (3) der Vereinssatzung bis zum 15. Dezember schriftlich an den Vorstand zu stellen. Ihre Stimmunterlagen erhalten Sie direkt vor Ort; um vorherige Anmeldung wird gebeten. Eine Übertragung des Stimmrechts ist im Rahmen des § 13 (4) der Vereinssatzung möglich. Nichtmitglieder sind als Gäste herzlich willkommen.

Unter `https://www.dante.de/veranstaltungen/dante2025/` finden Sie die Tagungsseite mit allen weiteren Informationen rund um die Veranstaltung. Neuigkeiten zur Tagung gibt es auch über die Vereinsmailingliste. Wir bitten Sie wegen der besseren Planbarkeit um eine frühzeitige Anmeldung über die Tagungswebsite. Diese wird voraussichtlich Anfang Januar freigeschaltet.

Falls Sie ein Tutorium oder einen Vortrag anbieten wollen, beachten Sie bitte den »Call for Presentations« auf S. 8.

Für Fragen, Wünsche und Anregungen schreiben Sie bitte ausschließlich per E-Mail an `dante2025@dante.de`.

Mit freundlichen Grüßen
Martin Sievers (Vorsitzender DANTE e.V.); Torsten-Karl Strempel (lok. Organisator)

Europäisches Raumflugkontrollzentrum (ESOC) in Darmstadt

Reise Reise, CC BY-SA 4.0 <https://creativecommons.org/licenses/by-sa/4.0>, via Wikimedia Commons

Beiträge gesucht (»Call for Presentations«)

Martin Sievers, Torsten-Karl Strempel

Wir möchten als Organisatoren natürlich ein spannendes und vielfältiges Vortragsprogramm für die Frühjahrstagung 2025 in Darmstadt anbieten können. Dazu sind wir allerdings auf Ihre und Eure aktive Unterstützung angewiesen.

Mögliche Themen für Einreichungen können sein:

- Erfahrungsberichte zum Einsatz von TEX bzw. Open-Source-Software,
- Einsatz von TEX in Lehre und Forschung an wissenschaftlichen Einrichtungen,
- Nutzung von TEX für den Satz von Facharbeiten bzw. anderer Abschlussarbeiten, Präsentationen etc. an (Hoch-)Schulen,
- Vorstellung spezieller Erweiterungen für den Einsatz in Beruf und Ausbildung,
- Beispiele aus der Praxis (beispielsweise Realisierung besonderer Anforderungen), eigene Klassen und Pakete,
- Einbinden von Schriften, Grafiken etc.,
- Typografie und ihre Umsetzung in TEX und Co.,
- Zusammenspiel von TEX mit anderen Dateiformaten (z. B. XML) und anderen (Open-Source-)Werkzeugen,
- die Entwicklung von TEX und Co. in den vergangenen Jahrzehnten,
- Mustererzeugung und Zeichnen mit Ti*k*Z oder PSTricks,
- Barrierefreie bzw. -arme PDF-Dokumente.
- …

Fühlen Sie sich angesprochen? Dann senden Sie bitte möglichst bis zum 28. Februar 2025 eine E-Mail mit folgenden Angaben an `dante2025@dante.de`:

- Name der Referentin/des Referenten,
- Titel und Art des Beitrags (Vortrag, Tutorium oder Lightning Talk),
- Zeitbedarf (*Tutorien* dauern im Allgemeinen 60 bis 90 Minuten (längere Tutorien sind möglich); für *Vorträge* beträgt die übliche Dauer 30 Minuten plus 10 Minuten für die anschließende Diskussion; *Lightning Talks* dauern maximal zehn Minuten),
- Zusammenfassung (max. 500 Wörter, ca. 0,5 bis 1,5 Seiten),
- evtl. benötigte Hilfsmittel (jenseits von Beamer und PDF-Viewer),
- evtl. Wünsche bzgl. der Vortragszeit.

Wir freuen uns über alle Einreichungen; es darf auch gerne Ihr erster »Auftritt« bei einer Tagung von DANTE e.V. sein.

Bericht von der TUG 2024 in Prag — Tag 1

Volker RW Schaa

Das 45. Treffen der TEX Users Group fand nach drei Online-Events wegen COVID zum zweiten Mal in Europa statt, dieses Mal in Prag in der Tschechischen Republik. Das Treffen fand vom 19.–21. Juli in Hotel Grandior statt, das fußläufig zur Altstadt gelegen ist.

Nach zögerlichem Anmeldestart hatten sich dann doch ca. 60 TEXies auf den Weg nach Prag gemacht, so dass nach dem sehr gut besuchten Treffen in Bonn (>90 Teilnehmer) wieder eine akzeptable Zahl von TEXies die Gelegenheit nutzten, neben den Vorträgen auch das Wiedersehen mit Freunden und Bekannten aus alten Tagen gebührend zu feiern.

Der Nachmittag vor der Konferenz (18. Juli) war für einen *PDF Developers' Workshop* reserviert, der für Paket- und Klassenentwickler gedacht war, um über den aktuellen Stand bei der Entwicklung von *Tagged and Accessible PDF* zu berichten, der helfen soll, die Barrierefreiheit von PDF-Dokumenten zu verbessern. Am Abend gab es einen Empfang mit Häppchen und Aperitifs im Tagungshotel sowie eine erste Begegnung der Tagungsteilnehmer, die sich im Anschluss in Grüppchen zu Restaurants und Bars in der Altstadt aufmachten.

Freitag, 1. Tag

Boris Veytsman (Foto siehe Seite 14), Vizepräsident der TUG, eröffnete die Tagung in Vertretung des Präsidenten der TUG, Arthur Rosendahl, der in der Vorwoche zum zweiten Mal Vater geworden war.

In seiner Begrüßung hob Boris die besondere Bedeutung der tschechischen TEXies hervor, die an der Förderung und Entwicklung von TEX großen Anteil haben. Hàn

Thế Thành, persönlich anwesend, hatte an der Masaryk Universität in Brno pdfTEX im Rahmen seiner Doktorarbeit entwickelt.

Der erste Redner war *Norbert Preining* mit seinem Vortrag »TEX (Live) at arXiv«. Norbert musste seine Teilnahme kurzfristig absagen und hatte daher eine Videopräsentation vorbereitet. arXiv ist der größte und älteste wissenschaftliche Preprint-Server der Welt und ein Verfechter der offenen Wissenschaft (Open Science, Open Source, Open Access). Beweise berühmter Theoreme wurden hier erstmals veröffentlicht, erste Artikel über LLM (OpenAI, Deepmind u. a.) und LIGO (Gravitationswellenforschung, Nobelpreis in Physik 2017) wurden auf arXiv publiziert. 90 % der Artikel liegen in LATEX/TEX vor. Seit seiner Gründung im Jahr 1991 enthält arXiv derzeit mehr als 2,4 Millionen Artikel und wächst beständig weiter.

Mit seiner über 30-jährigen Geschichte enthält arXiv TEX-Dokumente, die fast die gesamte Lebenszeit von TEX selbst umfassen. Bis heute versucht das Team, alle Dokumente neu zu kompilieren. Zu diesem Zweck werden mehrere Versionen von TEX-Distributionen vorgehalten, beginnend mit teTEX 2 bis hin zum derzeitigen TEX Live 2023. In seinem Vortrag berichtete Norbert über die Probleme, mit denen das Team konfrontiert ist, all diese Versionen am Laufen zu halten und mit der ständig wachsenden Anzahl von Paketen umzugehen, die häufig veraltet sind. Hinzu gesellen sich all die Besonderheiten, die mit der Unterstützung sehr alter Software verbunden sind. Die wesentlichen Unzulänglichkeiten betreffen den fehlenden Support für LuaTEX, XƎTEX, BibTEX oder biber (es müssen händisch aufbereitete `bbl`-Dateien hochgeladen werden) und makeindex. Die wichtigsten Ziele der nahen Zukunft sind der Support für LuaTEX und XƎTEX.

Der zweiter Sprecher an diesem Tag war *Mitch Gerrard*, Titel seines Vortrages »Holon Programming Regaining«. Die Holon-Programmierung war ein konzeptionelles Leitbild, das in den frühen 1970er Jahren von Pierre-Arnoul de Marneffe entwickelt wurde. Sein Ziel war es, Programmierern eine Möglichkeit zu geben, »strukturierte Programme« zu schreiben, die hauptsächlich Beschreibungen in natürlicher Sprache verwenden, die über einen »Synthesizer« durch Blöcke von ausführbarem Code ersetzt werden. Er hatte seine Schrift *Holon Programming: A Survey* 1973 unter Informatikern in Umlauf gebracht. Es wurde zu einer der Hauptinspirationen für das literarische Programmieren. Das Dokument wurde danach zu einem mythischen Buch, das nur in Zitaten von Knuth existiert.

Donald Knuth wurde durch diese Schrift zur Entwicklung des *Literate Programming* inspiriert. Er beschreibt den Ansatz von de Marneffe als »eine Möglichkeit, ein kompliziertes Programm in kleine Teile zu zerlegen. Um das komplizierte Ganze zu

verstehen, muss man dann nur noch die kleinen Teile verstehen und die Beziehung zwischen diesen Teilen und ihren Nachbarn.«

Mitch machte sich auf eine Such- und Rettungsmission dieses seltenen Buches, schrieb an Donald Knuth und erfuhr, dass er seine Ausgabe der Stanford Bibliothek geschenkt hatte. Dort war es aber nicht aufzufinden. Weitere Hinweise von Bibliothekaren führten schließlich zum einzigen erhaltenen Exemplar in Hannover, das ihm freundlicherweise zur Verfügung gestellt wurde. Jetzt ist dank Mitch eine kommentierte Ausgabe mit dem Brief von Donald Knuth (vom 1. April 1974!) auf `https://github.com/holon-scribe/holon-programming/` nach 50 Jahren wieder verfügbar.

Als nächstes sprach *Vincent Goulet* unter dem Titel »You (S)wove? Well (S)tangle now!« über das Thema *Literate Programming*. Das Konzept des *Literate Programming*, das 1984 von Donald Knuth entwickelt wurde, ist vielen in der TEX-Gemeinde bekannt. Kurz gesagt, handelt es sich um ein Programmierparadigma, bei dem ein Programm und seine Dokumentation in einer einzigen Datei zusammengefasst sind. Mit den Verfahren TANGLE wird der Quellcode und mit WEAVE die Dokumentation extrahiert. *Literate Programming* spielt heutzutage eine zentrale Rolle im wissenschaftlichen Computing für reproduzierbare Forschungszwecke: Anstatt in einem Bericht hart kodiert zu werden, werden Ergebnisse und Grafiken mit Hilfe des Computercodes »eingewebt«. Im `R`-Ökosystem sind SWEAVE und KNITR weit verbreitet, um auf diese Weise Dokumente aus `R`-Code zu erstellen. In seinem Vortrag beleuchtete Vincent die vielleicht weniger bekannte Komponente im wissenschaftlichen Rechnen: den Schritt TANGLE. Er zeigte einen Anwendungsfall, in dem eine geschickte Kombination aus `Sweave` und `Stangle` die effiziente Verwaltung einer Reihe von Aufgaben und Lösungen ermöglicht (`Stangle` inside `Sweave`).

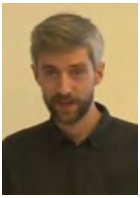
Martin Ruckert hielt den ersten Vortrag nach der Kaffeepause und beleuchtete das Thema des Laufzeitverhaltens von TEX in seinem Vortrag »Profiling TEX input files«. Ein Profiler ist ein Standardwerkzeug für Programmierer, das verwendet werden kann, um herauszufinden, wie sich die Laufzeit eines Programms verhält, indem es die Ausführungszeit einzelner Codezeilen abbildet. Wendet man ein solches Standardwerkzeug auf einen TEX-Lauf an, erfährt man vielleicht, dass es 28 % mit Zeilenumbrüchen verbringt, 20 % in der Hauptschleife, 19 % für die Speicherverwaltung und 18 % für die Erzeugung der Ausgabedatei. Diese Informationen können von Nutzen sein, wenn man eine TEX-Engine implementieren will; für jemanden, der ein TEX-Paket entwickelt, sind sie nahezu nutzlos.

Um diese Situation zu beheben, hat Martin einen Profiler für TEX implementiert,

der die Nutzung der Laufzeit einzelnen Zeilen der TEX-Eingabedateien zuordnen kann. Damit erlaubt es dem Entwickler (nicht dem Autor) zu untersuchen, welche Optimierungsmöglichkeiten der geschriebene Code bietet (und auch, welche nicht) und ob eine Optimierung erfolgreich war.

Besondere Herausforderungen bei der Entwicklung des Profilers waren zum einen Hardware-Probleme (verlässliche Zeitmessung, Interrupts und Time-sharing, Verhalten der CPU-Cores u. a.), zum anderen Probleme, wie man für jedes Token den Dateinamen, die Zeilennummer und den Makronamen ermittelt und speichert. Für die Tests wurden vier Dokumente benutzt, die sich in Seitenzahl (3 bis 1130), Makro-Nutzung (niedrig, mittel, hoch) und Quelle (plainTEX, LATEX und `cweb`) unterschieden.

Das Fazit der Entwicklung zeigte auf, dass der Profiler (noch) nicht perfekt ist, er Makroentwicklern aber aufzeigt, wo Raum für Optimierungen ist (und wo nicht). Die Software ist Open Source und auf `GitHub` verfügbar.



Anschließend referierte *Tyge Tiessen* zum Thema »Rewriting TEX today« darüber, dass er sich als Kryptograf zum »Zeitvertreib« ein, wie er es empfand, mittelgroßes (Neben-)Projekt suchte und sich dabei an die Reimplementierung von TEX82 in `Rust` wagte, obwohl er weder Experte in TEX noch in `Rust` war. TEX82 wurde vor vierzig Jahren geschrieben. Begrenzten Speicherressourcen sowie Bedenken bezüglich der Portabilität und Compilerbeschränkungen setzten der ursprünglichen Implementierung enge Grenzen. Einige der Folgen davon sind die handgeschriebene dynamische Speicherverwaltung, die allgegenwärtige Verwendung globaler Variablen und die manuelle Verwaltung von Zeichenketten in TEX82 (Poolfile).

Tyge beschreibt seinen Ansatz zur Reimplementierung folgendermaßen: Begonnen wurde mit der 1:1-Übersetzung des ursprünglichen `web`-Codes, die Korrektheit des Codes wurde jeweils durch `TRIP`- und andere Tests sichergestellt und dann wurde der Code Stück für Stück in `Rust`-Konstrukte umgesetzt. Bisweilen war es schwierig, alle Nuancen des Codes zu verstehen, denn das Lesen eines Eingabe-Tokens erfordert die Kenntnis des aktuellen Kategoriecodes und je nachdem, wo das Token gelesen wird, können fehlerhafte Annahmen erhebliche Fehler verursachen.

Der aktueller Stand des Projekts besteht den `TRIP-Test3`: Es erzeugt eine identische Ausgabe (abgesehen von Zeitstempeln und Versionsstrings), es enthält keine globalen Variablen und keinen »unsafe« Code mehr. Die meisten bestehenden Beschränkungen wurden aufgehoben: Anzahl der Kontrollsequenzen, Anzahl der Register und Strings. Aber: es ist etwa um den Faktor 2 langsamer, hat keine Kpathsea-Integration (nur hart kodierte Verzeichnisse), keine funktionierenden Zeitstempel und ist nicht portabel, da nur für Linux implementiert. Es bietet keine

PDF-Ausgabe, keine »e-TEX-Erweiterungen«, wenig Dokumentation, zumeist nur als Verweise auf *TEX: The Program*. Der Code ist auf `GitHub` verfügbar.

Es folgte der Vortrag von *Didier Verna* mit dem Titel »A large scale format compliance checker for TEX font metric files«. Auf den ersten Blick schien mir die Arbeit aus der Zeit gefallen, hatte ich mich doch weitestgehend auf TrueType- und OpenType-Fonts umgestellt und keinen Gedanken mehr an Type 3 & 1-Fonts und deren TFM-Dateien verschwendet. Erst die riesige Anzahl der vorhandenen (und überprüften) TFM-Dateien bewegte mich zum Umdenken bezüglich des Nutzens eines solchen Tools. Dies insbesondere auch für die Archivierung oder Regeneration alter Dokumente, ein Problem, dass Norbert in seinem Vortrag angesprochen hatte.

Didier stellte auch gleich zu Beginn klar, dass seine Motivation mehr von der Neugier als von der Notwendigkeit herrührte und dass das Projekt einfach zu realisieren war. Als Teil von `ETAP`, einer experimentellen Plattform für Schriftsatzalgorithmen an der Grande École EPITA, wurde ein Parser für TEX-Font-Metrikdateien entwickelt, einfach `TFM` genannt. Ein robuster Parser für ein offizielles Datenformat muss in der Lage sein, mit allen Arten von Konformitätsproblemen mit unterschiedlichen Schweregraden umzugehen, die von einfachen Warnungen bis hin zu Fehlern reichen, die nicht korrigiert werden können. `TFM` bietet nicht nur eine reichhaltige (hoffentlich erschöpfende) Sammlung von Fehlern, sondern auch einen leistungsstarken Wiederherstellungsmechanismus. Er soll es ermöglichen, während der Analyse so weit wie möglich Fehler im laufenden Betrieb zu beheben oder problematische Daten zu verwerfen.

Als Nebeneffekt ist es möglich, `TFM` als Validierungswerkzeug und nicht zum Laden von Schriftinformationen zu verwenden. Der Exception-Handler zeigt problematische Situationen in den Objekten (im »objektorientierten« Sinne) an. Diese werden während der Ausführung gesammelt, bis die Analyse abgeschlossen ist. Diese Objekte können wiederum zur Erstellung eines vollständigen Konformitätsberichts für die analysierte Datei verwendet werden.

Der Prozess wurde für die gesamte TEX Live-Distribution automatisiert, was zur Validierung von fast 80.000 Schriften geführt hat (siehe `https://texlive.info/tfm-validate/`). Dabei wurden 770 Fonts gefunden, die nicht den Regeln entsprechen. Also viele nicht konforme Fonts, aber nur 2 wirklich unbrauchbare (»Fix Word Overflow«), die TEX als defekt zurückweist. Eine mögliche Perspektive ist die Unterstützung bei der Analyse anderer Formate wie `OpenType`.

Als nächstes sprach *Boris Veytsman* unter dem Titel »Packing *Arsenal* fonts for XeLaTeX and LuaLaTeX« über die Erfahrungen, die er mit dem Prozess für die Schriftpaketierung, bei der Verwendung von `fontspec` und `expl3` gemacht hat.
2011 wurde der ukrainische Schriftdesign-Wettbewerb »Mystetsky Arsenal« (`http://www.ukrainian-type.com/about`) zu einem von Andrij Shevchenko entworfenen Font ausgeschrieben. Ziel des Wettbewerbs war es, eine moderne, praktische Schrift auf der Grundlage ukrainischer Traditionen zu entwerfen. Die Schrift, die den Wettbewerb gewann, zeichnet sich durch ihre Klarheit und sauberen Formen aus. *Arsenal* ist eine Halbgroteske (englisch »Humanist«) mit traditionellen Formen.

Die Veröffentlichung der Schrift unter einer freien Lizenz war eine der Wettbewerbsbedingungen. Dies erwies sich als Glücksfall, denn die Lizenz (SIL Open Font License) ermöglichte es, den ursprünglichen Entwurf von Alexei Vanyashin, Nhung Nguyen und Marc Foley erweitern zu können (siehe `https://github.com/alexeiva/Arsenal`). Die Schrift unterstützt nun zahlreiche Sprachen mit lateinischem und kyrillischen Alphabet, hat echte Kapitälchen, historische Formen, »Small Caps« und mit ihren Schriftschnitten `Swash`, `Swash Italic`, `Swash Bold` und `Swash Bold Italic` einen kompletten Satz von Schwungschriftzeichen (Swash).

Boris hat ein Paket für die Verwendung der *Arsenal*-Fontfamilie in LaTeX (nur für die Unicode-Engines) entwickelt und eine experimentelle Mathematikunterstützung hinzugefügt. Er präsentierte drei Varianten von *Arsenal* mit `KpSans` und `Iwona`. Bei der ersten wurden nur in der Mathematik fehlende Buchstaben aus `KpSans` benutzt. Das funktioniert nicht korrekt bei XeTeX, auffällig sind die Unterschiede in den Proportionen zwischen Buchstaben und Zahlen im Vergleich zu z. B. griechischen Buchstaben. Die zweite Variante (Standard bei XeTeX) benutzt die `Iwona` (Iwona-CondLight) für mathematische Zeichen, allerdings kamen im Beispiel die Zahlen, einige Klammern und relationale Operatoren aus den `cmr`- respektive `msbm`-Fonts. Im dritten Beispiel sind alle Zeichen des Mathematik-Modus aus der `KpMath-Sans`.

Zur Überraschung von Boris wählte die französischsprachige TeX User Group Gutenberg *Arsenal* als Schrift für die Novemberausgabe 51/2023 von *Lettre Gutenberg*. Dies ist ein eindrucksvoller Beweis dafür, dass sich die Schriftfamilie auch für solche Zwecke eignet.

*Vít Starý Novotný*s Vortrag mit dem Titel »Markdown themes in practice« drehte sich um die Konfiguration von benutzerdefinierten Markdown-Stilen für TeX. Obwohl TeX einen perfekten Textsatz produziert, ist seine Makrosprache für viele Autoren gewöhnungsbedürftig. Sie können ihre Texte in der simplen `Markdown`-Notation eingeben und die Ausgabe und Darstellung/Satz mittels TeX durch

Stylesheets resp. *themes* anpassen.

Vît zeigte in seinem Vortrag den aktuellen Stand der Markdown-Themes am Beispiel von LaTeX-Vorlagen, die er für das *International Software Testing Qualifications Board* (`ISTQB`) entwickelt hatte. In diesem Projekt entwickelte er die LaTeX-Dokumentenklasse `istqb` und sechs Markdown-themes. Die Klasse implementiert Design und LaTeX-Markup. Die zugehörigen »themes« heißen `istqb/sample-exam/*` und sind in der Sprache `expl3` geschrieben.

Als nächster Redner sprach *Wim Obbels* unter dem Titel »*Ximera* Interactive Math Educational Resources for All: From LaTeX source code to PDF, HTML and beyond« über die Erstellung interaktiver Unterrichtsmaterialien für Online und Druck.
`Ximera` ist eine Open-Source-Plattform zur Erstellung interaktiver Online-Kurse unter Verwendung von LaTeX als Quellcode. Es unterstützt das Verfassen und Veröffentlichen von interaktiven Bildungsinhalten, Arbeitsblättern, Beurteilungen, Lehrbüchern und Online-Kursen sowohl in HTML (Online) als auch PDF (Online und Druck). Ursprünglich an der Ohio State University (`OSU`) entwickelt, nutzen inzwischen rund ein Dutzend Institutionen die Ximera-Materialien. Die Backend-Entwicklung des Projekts hat sich von der `OSU` auf Lehrkräfte an der Universität von Florida und der `KU` Leuven verlagert.

Im Vortrag wurden die grundlegenden Ideen der Funktionsweise von Ximera erläutert. Es wurde demonstriert, wie die derzeit verfügbaren Kurse interaktive Fragen, Applets von `youtube`, `Desmos` oder `geogebra`, Hyperlinks usw. einbinden und wie sich diese, basierend auf TeX4ht, ziemlich reibungslos zwischen HTML und PDF übersetzen lassen.

Nach der Installation des Ximera-Pakets von CTAN kann man sofort PDF-Versionen erzeugen. Um eine Online-Version zu veröffentlichen, wird eine zusätzliche Build-Umgebung (`xake`) benötigt, die die Veröffentlichung von Kursen auf einem öffentlichen Server unter `https://ximera.osu.edu` oder optional auf einem selbst gehosteten Server erlaubt. Ximera kann in Lernmanagementsysteme integriert werden, sammelt nützliche Lernanalysen und kann die Ergebnisse als Notenspiegel zurückgeben. Es wird aktiv weiterentwickelt und erhält zusätzliche Funktionen, mehr Beispiele und Tutorials, Optionen für die Gestaltung von Kursen sowohl online als auch in PDF. Außerdem wird an einem Docker-Setup für eine vereinfachte Bereitstellung und sogar serverloses Setup mit Kompilierung-im-Browser gearbeitet.

Zum Abschluss der Vorträge an diesem Tag sprach *Sarah Lang* über den Einsatz von LaTeX in den digitalen Geisteswissenschaften. Titel ihrer Präsentation: »LaTeX in the Digital Humanities«. Dieser Vortrag erforscht den Schnittpunkt von LaTeX-Satz und den digitalen Geisteswissenschaften. Genauer gesagt geht es um die Frage, warum LaTeX so selten in den digitalen Geisteswissenschaften verwendet wird, trotz der klaren Vorteile und Anwendungen. Dieser Vortrag zielt zum einen darauf ab Gründe zu verstehen, warum LaTeX in den digitalen Geisteswissenschaften nicht häufiger eingesetzt wird und andererseits werden drei Beispiele für relevante Anwendungsfälle gezeigt, um den Wert von LaTeX für die (digitalen) Geisteswissenschaften zu illustrieren.

Konferenz-Proceedings, die die Einreichung kompletter Papiere erfordern, wie z. B. die *Computational Humanities Conference*, werden in den Digital Humanities immer häufiger und bieten daher einen guten Grund, sich mit den notwendigen LaTeX-Kenntnissen vertraut zu machen. Dieser Vortrag stellt die von *Quinn Dombrowski* vertretene Auffassung in Frage, dass das Erlernen von LaTeX Geisteswissenschaftlern zu viel abverlangt. Geisteswissenschaftler, auch solche mit minimalem technischem Hintergrund, können LaTeX auf dem erforderlichen Niveau erlernen, um ihre Beiträge effektiv zu formatieren, indem sie es im Wesentlichen als Auszeichnungssprache verwenden.

Der zweite und wichtigste Anwendungsfall für LaTeX in den digitalen Geisteswissenschaften ist die digitale wissenschaftliche Bearbeitung. In diesem Bereich gibt es eine beträchtliche Gemeinschaft. Man könnte sogar sagen, dass das digitale wissenschaftliche Editing eines der Kernarbeitsfelder für digitale Geisteswissenschaftler ist. Digitales wissenschaftliches Editing ist ein Bereich, in dem das Fachgebiet ursprünglich florierte, obwohl die jüngsten Entwicklungen im Bereich des Deep Learning und des Large Language Models den Teilbereich der »Computational Humanities« mehr und mehr dominieren. Digitales wissenschaftliches Editieren wird eine zentrale Aufgabe und Technologie innerhalb der Digital Humanities bleiben; daher wird LaTeX auch weiterhin für die Digital Humanities eine Relevanz haben.

Trotz früher Vorhersagen, dass digitale Medien physische Bücher überflüssig machen, stellen wir im Jahr 2024 fest, dass dieser Wandel noch nicht eingetreten ist und es unwahrscheinlich erscheint, dass dies in naher Zukunft der Fall sein wird. Im Gegenteil, der Wert des Buches als materielles Objekt und als Symbol für kulturelles Kapital ist gestiegen, insbesondere bei einem jüngeren Publikum auf Plattformen wie BookTok. Dieser Trend mag zwar für die Wissenschaft an sich irrelevant erscheinen, unterstreicht aber einen wichtigen Punkt: Physische Bücher behalten ihre Bedeutung bis zum heutigen Tag und werden weiterhin von Textwissenschaftlern für verschiedene Aufgaben, wie z. B. vertieftes Lesen, genutzt. Diese Vorliebe für

gedruckte Materialien macht deutlich, wie wichtig es ist sicherzustellen, dass digitale geisteswissenschaftliche Ressourcen, wie digitale wissenschaftliche Editionen, bei Bedarf in gedrucktes Material übersetzt werden können. Die Umwandlung digitaler wissenschaftlicher Editionen vom `TEI-XML`-Format in LaTeX ermöglicht es uns, Druckversionen von digitalen Editionen zu erstellen, selbst dann, wenn die Ausgabe nicht in erster Linie in gedruckter Form veröffentlicht werden sollte.

Durch die Umwandlung von `TEI`-Daten mit `XSLT`, möglicherweise sogar unter Verwendung großer Sprachmodelle zur Generierung von LaTeX-Code, können Wissenschaftler mit geringem Aufwand hochwertige gedruckte Materialien erstellen. Diese Funktionalität ist für die Gemeinschaft der digitalen Geisteswissenschaften von entscheidender Bedeutung, da sie der häufigen Vorliebe für zugängliche, druckbare Formate Rechnung trägt, die besonders in der Lehre nützlich sind. Daher besteht weiterhin eine erhebliche Nachfrage nach gedruckten Exemplaren, so für detaillierte Studien oder zur Verwendung im Unterricht. Das `reledmac`-Paket ist ein besonders nützliches Werkzeug für Digital Humanists.

Der Nutzen von LaTeX bei der Verwaltung komplexer geisteswissenschaftlicher Dokumente, wie z. B. archäologischer Kataloge, muss hervorgehoben werden. In Bereichen wie der Archäologie ist es üblich, dass umfangreiche Kataloge von Objekten und Funden integraler Bestandteil von größeren Forschungsprojekten, wie z. B. Dissertationen, sind. Solche Kataloge enthalten oft zahlreiche Bilder und können bei der Verwaltung mit Standard-Textverarbeitungsprogrammen aufgrund ihres Umfangs, bei Indexerstellung, Abbildungs- und Inhaltsverzeichnissen, sehr schnell unhandlich werden. Auch wenn diese Anwendung nicht ausschließlich im Bereich der digitalen Geisteswissenschaften zu verorten ist, unterstreicht sie die Relevanz von LaTeX in den breiteren geisteswissenschaftlichen Disziplinen, wie bereits im LaTeX Ninja-Blog diskutiert wurde.

Oliver Kopp, *Marei Peischl* und *Marcel Krüger* bestritten den letzten Block des Tages mit dem Tutorium über »Creation of LaTeX documents using a cloud-based pipeline«. Dieses Tutorial soll die Verwendung von `git` sowie `GitHub/GitLab` Pipelines für die Entwicklung von LaTeX-Dokumenten zeigen.

Die meisten LaTeX-Benutzer kompilieren auf ihrem lokalen Rechner oder über Online-Dienste wie `Overleaf`. Hier wurde eine dritte Möglichkeit präsentiert: der Build-Server. Diese Option ist besonders interessant, wenn man `git` verwendet. Im Tutorial wurde gezeigt, wie man von einer `git`-Installation zu seinem LaTeX-Dokument gelangt, das von einem Build-Server kompiliert wurde. Zu Beginn wurden die Konzepte von `git` und `GitHub/GitLab` erklärt, bevor es an den praktischen Teil ging.

Bretter, die die Welt bedeuten

LaTeX und Schulphysik 8: Atom- und Kernphysik

Keno Wehr

Der achte Teil der Artikelreihe zur Schulphysik beschäftigt sich mit Schreibweisen und grafischen Darstellungen zur Atom- und Kernphysik. Er stellt Befehle aus den Paketen `bohr`, `chemfig`, `chemformula`, `chemmacros`, `elements`, `pgf-PeriodicTable`, `pgf-spectra` und `textgreek` vor.

Atome, Moleküle und Ionen

Zur Darstellung von Elementsymbolen ist in der Regel keine besondere Auszeichnung erforderlich. Da diese aufrecht zu setzen sind [2, S. 9], sollten sie im gewöhnlichen Textmodus (nicht im Mathematikmodus) gesetzt werden. Auch für das Wassermolekül reichen Standardbefehle im Prinzip noch aus.

```
Hg\\
NaCl\\
H\textsubscript{2}O
```

Hg
NaCl
H_2O

Wenn mehrfach hoch- und tiefgestellte Zeichen auftreten, ist die Verwendung des Befehls `\ch` aus dem `chemformula`-Paket vorzuziehen.

```
\usepackage{chemformula}
```

```
\ch{H2O}\\
\ch{CH4}\\
\ch{Na+}\\
\ch{Cl-}\\
\ch{Ca^{2-}}
```

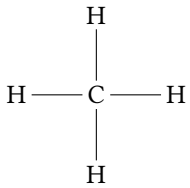
H_2O
CH_4
Na^+
Cl^-
Ca^{2-}

Strukturformeln von Molekülen können mit dem (sehr mächtigen) Paket `chemfig` gezeichnet werden. Einfachbindungen werden als Bindestrich eingegeben, Doppelbindungen als Gleichheitszeichen; mittels eines optionalen Arguments kann der

Bindungswinkel angegeben werden. Verzweigungen sind in runde Klammern zu setzen.

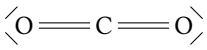
```
\usepackage{chemfig}
```

```
\chemfig{H-C(-[:90]H)(-[:-90]H)-H}
```



Freie Elektronenpaare werden mit dem `\charge`-Befehl platziert.

```
\chemfig{\charge{135=\|,-135=\|}{O}=C=\charge
 ↪{45=\|,-45=\|}{O}}
```



Der Physiklehrer wird die Möglichkeiten des Pakets nicht annähernd ausschöpfen, doch mag er immerhin eine Darstellung des Wassermoleküls wie die folgende zur Erklärung der Polarität nutzen wollen.

```
\chemfig{H<[:45]\charge{45=\|,135=\|}{O}>[:-45]H}
```

Aufrechte griechische Kleinbuchstaben

Bei der Bezeichnung der charakteristischen Linien der Röntgenstrahlung (K_α-Linie, K_β-Linie etc.) sowie der radioaktiven Strahlungsarten (α-, β- und γ-Strahlung) müssen die griechischen Kleinbuchstaben aufrecht gesetzt werden, da sie in diesen Fällen keine physikalischen Größen bezeichnen [2, S. 9 f.]. Die Verwendung der Befehle `\alpha`, `\beta` etc. bei Wechsel in den Mathematikmodus verbietet sich daher.

Die voreingestellte und häufig verwendete Schriftart *Latin Modern Roman* verfügt nicht über aufrechte griechische Kleinbuchstaben. Zu dieser Schriftart passende griechische Buchstaben macht jedoch das Paket `textgreek` durch die Befehle `\textalpha`, `\textbeta`, … verfügbar.

```
\usepackage{textgreek}
```

```
K\textsubscript{\textalpha}-Linie\\
\textbeta-Zerfall\\
\textgamma-Strahlung
```

K_α-Linie
β-Zerfall
γ-Strahlung

Nicht verwendet werden sollte `textgreek`, wenn eine Schrift zum Einsatz kommt, die über eigene griechische Buchstaben verfügt. Dies betrifft beispielsweise die freien OpenType-Schriften *Libertinus Serif*, *Linux Libertine O*, *TeX Gyre Pagella*, *TeX Gyre Schola* und *TeX Gyre Termes*. Das Problem liegt in diesem Fall in der Eingabe der griechischen Buchstaben, was einen Spezialfall der Eingabe von Sonderzeichen darstellt, wofür auf [3] verwiesen sei. Je nach Eingabemethode kann es hilfreich sein, die Unicode-Codepunkte von α ($03B1_{16}$), β ($03B2_{16}$) und γ ($03B3_{16}$) zu kennen.

```
\usepackage{libertinus-otf}
```

```
K\textsubscript{α}-Linie\\
β-Zerfall\\
γ-Strahlung
```

K_{α}-Linie
β-Zerfall
γ-Strahlung

Wenn die direkte Eingabe scheitert, kann man sich mit dem Befehl `\symbol` behelfen.

```
\usepackage{pagella-otf}
```

```
K\textsubscript{\symbol{"03B1}}-Linie\\
\symbol{"03B2}-Zerfall\\
\symbol{"03B3}-Strahlung
```

K_{α}-Linie
β-Zerfall
γ-Strahlung

Wenn das Dokument mit `latex` oder `pdflatex` übersetzt wird, funktionieren die beiden letztgenannten Methoden nicht; diese Compiler sind aber m. E. ohnehin als veraltet zu betrachten und durch `xelatex` oder `lualatex` zu ersetzen.

Schalenmodell und Elektronenkonfiguration

Die Verteilung der Elektronen nach dem Schalenmodell, das im Oberstufenunterricht typischerweise als Grundlage zur Erklärung des charakteristischen Röntgenspektrums dient, kann mit Hilfe des Pakets `bohr` veranschaulicht werden.[1] Der Befehl `\bohr` erwartet zwei Argumente: die Anzahl der Elektronen und eine Beschriftung, typischerweise das Elementsymbol (Abb. 1).

```
\bohr{1}{H}\quad
\bohr{10}{Ne}\quad
\bohr{29}{Cu}
```

[1] Der Name des Pakets bezieht sich offenbar nicht auf die bekannte semiklassische Atomtheorie Niels Bohrs von 1913, in der Bohr noch keine korrekten Aussagen über die Belegung der erlaubten Bahnen mit Elektronen machen konnte, sondern auf spätere Erweiterungen des Modells, die wissenschaftshistorisch nicht Bohr allein zugeschrieben werden können.

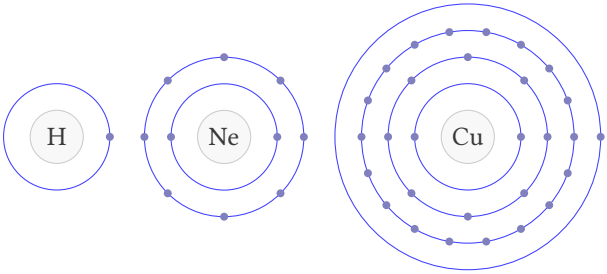


Abb. 1: Paket `bohr`: ein Wasserstoff-, ein Neon- und ein Kupferatom im Schalenmodell

Das Paket bietet auch Optionen, die dafür sorgen, dass die Elektronenanzahl oder das Elementsymbol automatisch ermittelt wird, wenn das entsprechende Argument leer bleibt.

Passend dazu erlaubt das Paket `elements` die Darstellung der zugehörigen Elektronenkonfigurationen:

```
\usepackage{elements}
```

```
Wasserstoff: \elconf{H}\\
Neon: \elconf{Ne}\\
Kupfer: \elconf{Cu}
```

Wasserstoff: $1s^1$
Neon: $1s^2 2s^2 2p^6$
Kupfer: $1s^2 2s^2 2p^6 3s^2 3p^6 3d^{10} 4s^1$

Periodensystem

Das Periodensystem der Elemente sowie beliebige Ausschnitte daraus können mit dem Paket `pgf-PeriodicTable` (man beachte die Binnenmajuskeln) dargestellt werden. Eine deutschsprachige Lokalisierung ist vorhanden und kann über eine Paketoption gewählt werden.[2]

```
\usepackage[de]{pgf-PeriodicTable}
```

Mit dem bloßen Befehl `\pgfPT` wird ein vollständiges Periodensystem mit Standardeinstellungen ausgegeben (Abb. 2).

```
\pgfPT
```

Mit Hilfe zahlreicher Optionen kann das Aussehen des Periodensystems verändert werden. Unser erstes Anliegen ist es, die zwar offiziell empfohlenen, aber didaktisch

[2] Die Lokalisierung ist zur Zeit noch unvollständig, da für alle gewählten Sprachen ein Punkt als Dezimaltrennzeichen verwendet wird. Der Paketautor hat Abhilfe zugesagt.

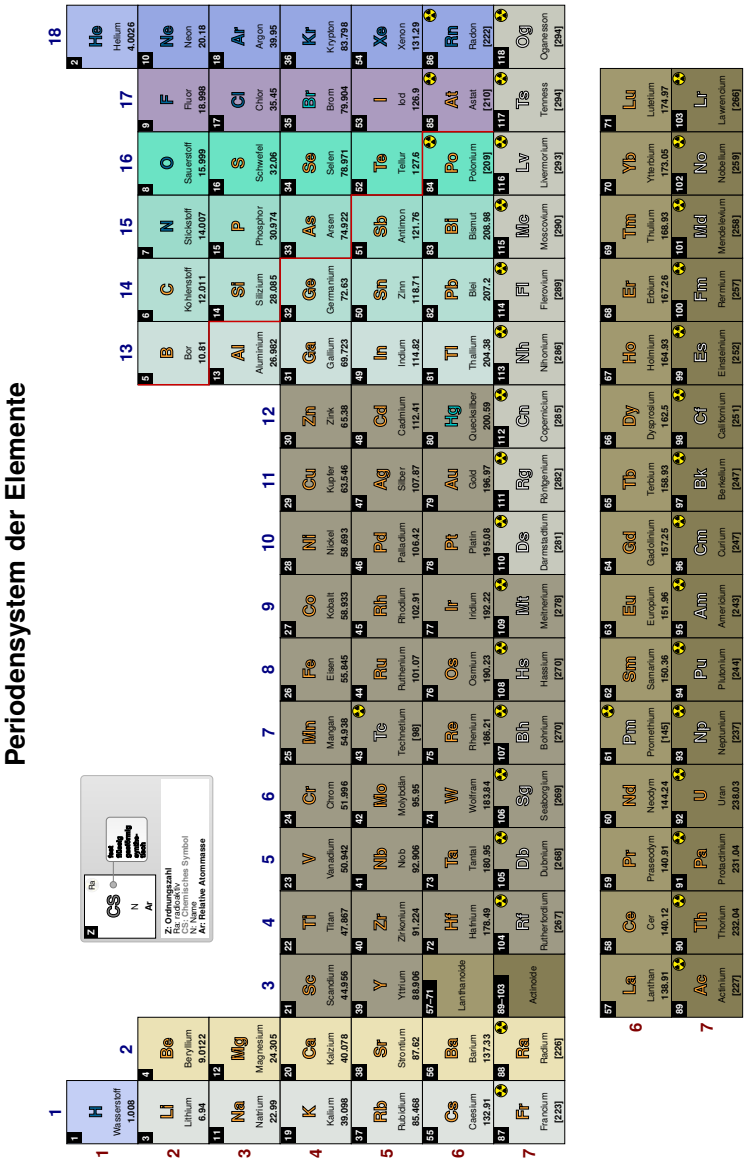


Abb. 2: Paket `pgf-PeriodicTable`: vollständiges Periodensystem der Elemente mit Standardeinstellungen (verkleinerte Darstellung)

wenig sinnvollen Gruppennummern von 1 bis 18 durch solche von I bis VIII zu ersetzen. Von den möglichen Werten der Option `group numbers` entspricht am ehesten `CAS`, was für *Chemical Abstract Service* steht, unserem Anliegen. Außerdem wählen wir mit `csMNM` (*color scheme metal/non-metal*) ein Farbschema, das Metalle, Halbmetalle und Nichtmetalle unterscheidet, und entfernen mit `show MNM line=false` die rote Linie, die lediglich die ungefähre Grenze zwischen Metallen und Nichtmetallen markiert (Abb. 3).

```
\pgfPT[group numbers=CAS,csMNM,show MNM line=false]
```

Wenn man nur einen Ausschnitt aus dem Periodensystem benötigt, kann man der Option `Z list` eine Liste der Ordnungszahlen der gewünschten Elemente übergeben. Beispielsweise sind für die Halbleiterphysik die Elemente aus der 3. bis 5. Hauptgruppe ausreichend (Abb. 4).

```
\pgfPT[Z list={5,6,7,13,14,15,31,32,33,49,50,51},group numbers=CAS,csMNM,show MNM
 ↪ line=false,show title=false,show legend=false]
```

Für die Liste der Ordnungszahlen, die an `Z list` übergeben wird, kann die Pünktchen-Konvention von Ti*k*Z verwendet werden: `{1,...,20}` steht für die Elemente 1 bis 20, `{1,3,...,25}` für die Elemente mit ungeraden Ordnungszahlen bis 25 und `{1,3,4,...,9}` für die Elemente 1 und 3 bis 9.

Standardmäßig wird als einzige Elementeigenschaft die relative Masse (mittlere Atommasse in atomaren Masseneinheiten) ausgegeben. Daneben bringt `pgfPeriodicTable` auch Daten zu Atomradien (drei verschiedene Arten), erster Ionisierungsenergie, Elektronegativität, Elektronenaffinität, Oxidationszahlen, Schmelz- und Siedetemperatur (in °C und K), Elektronenkonfiguration, Dichte, spezifischer Wärmekapazität, Wärmeleitfähigkeit, Gitterstruktur, Entdeckungsland und -jahr und Emissionsspektrum mit, die bei Bedarf ausgegeben werden können. Für die Ionisierungsenergie wird allerdings nur die im Physikunterricht ungebräuchliche Einheit kJ/mol unterstützt.

Die Auswahl der darzustellenden Elementeigenschaften erfolgt über sogenannte Zellstile (*cell styles*). Einige wenige Zellstile sind vordefiniert, die aber kaum alle Bedürfnisse abdecken. Um die auszugebenden Daten individuell auswählen zu können, muss ein eigener Zellstil definiert werden. Die Syntax dazu erfordert etwas Einarbeitung und wird in der Paketanleitung im Abschnitt »Designing cells with `\pgfPTbuildcell`« beschrieben. Wir definieren als Beispiel einen Zellstil, der die Atommasse, den Atomradius, die Elektronegativität und die Elektronenkonfiguration ausgibt (Abb. 5).

```
\pgfPTbuildcell(7,2)[(1;1;Z),(1;2;radio),(2-3;1-2;CS),(4;1-2;name),(5;1-2;Ar),
 ↪(6;1;R),(6;2;eneg),(7;1-2;eConfign)]
```

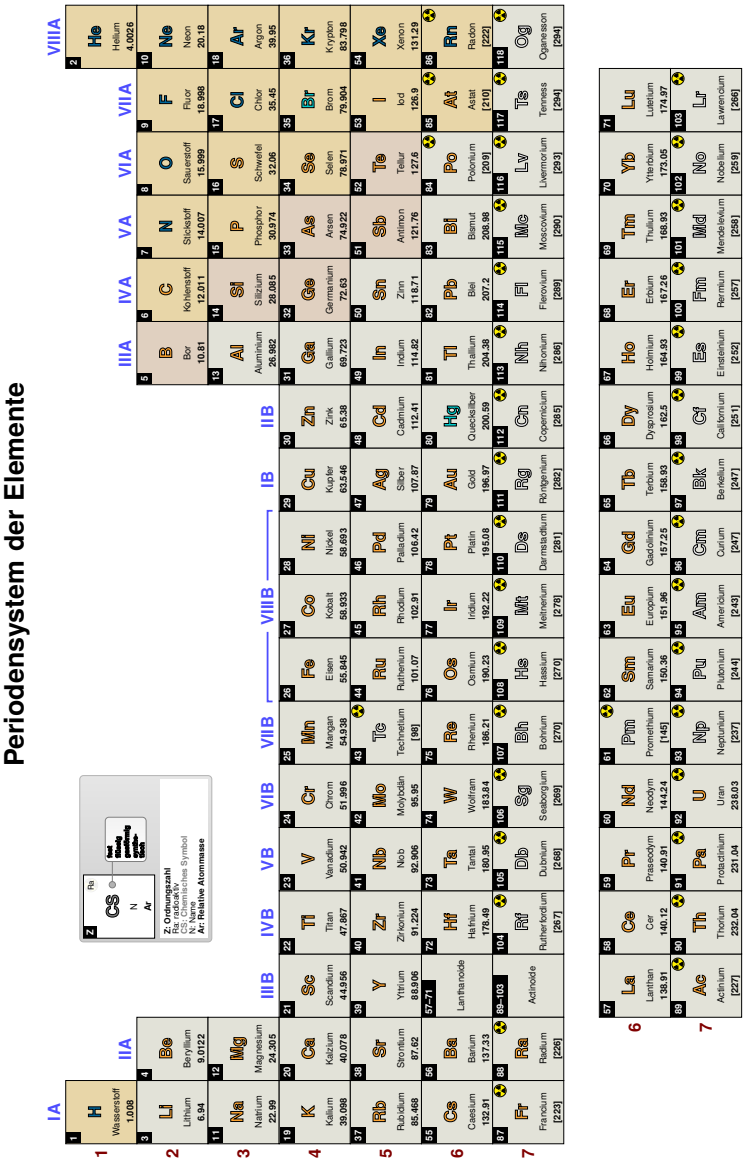


Abb. 3: Paket `pgf-PeriodicTable`: vollständiges Periodensystem der Elemente mit dem Farbschema `csMNM` (verkleinerte Darstellung)

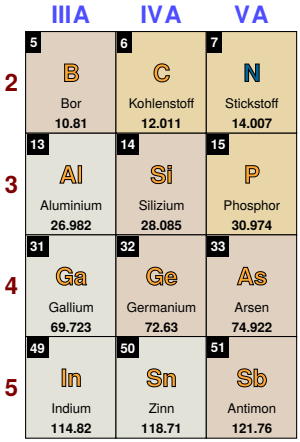


Abb. 4: Paket `pgf-PeriodicTable`: für die Halbleiterphysik relevanter Ausschnitt aus dem Periodensystem

```
\pgfPT[Z list={1,3,4,11,12,19,20,...,30,37,38,...,48,55,56,72,73,...,80,87,88,
 ↪104,105,...,112},group numbers=CAS,show title=false]
```

Nicht verschwiegen werden soll, dass die didaktischen Probleme, die mit der Vermischung von Atom- und Stoffeigenschaften in herkömmlichen Darstellungen des Periodensystems verbunden sind, auch von der Darstellungsweise von `pgf-PeriodicTable` gefördert werden [1].

Atomspektren und Energieniveauschemata

Die Darstellung von Emissions- und Absorptionsspektren macht das Paket `pgf-spectra` möglich. Die zugehörigen Daten für Atome und ihre Ionen bringt das Paket gleich mit, sodass in der Regel keine Wellenlängen, sondern nur das Element (Option `element`) und ggf. die Ladung (Option `charge`) angegeben werden müssen (Abb. 6).

```
\pgfspectra[element=H]\\
\pgfspectra[element=He]\\
\pgfspectra[element=He,charge=1]
```

Dargestellt wird standardmäßig der sichtbare Bereich von 380 nm bis 780 nm. Eine Skala kann mit geeigneten Optionen ergänzt werden.

```
\pgfspectra[element=H,axis,axis ticks=3,axis color=white,axis font color=black]
```

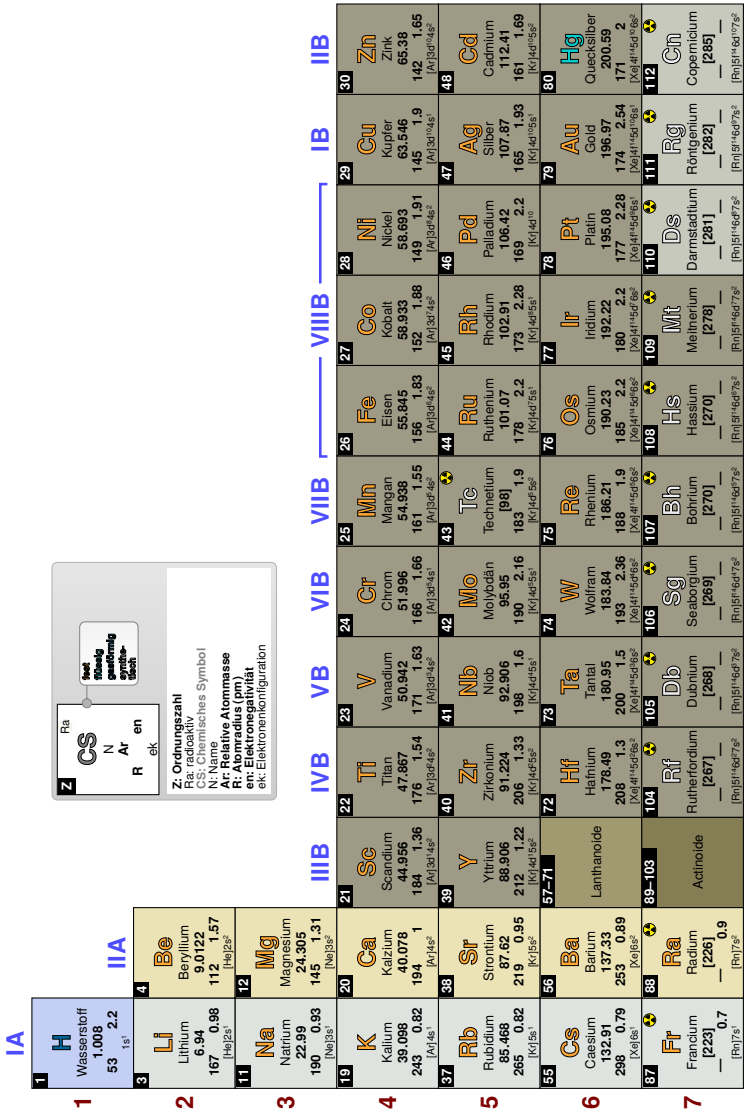


Abb. 5: Paket `pgf-PeriodicTable`: Ausschnitt aus dem Periodensystem mit Atommassen, Atomradien, Elektronegativitäten und Elektronenkonfigurationen

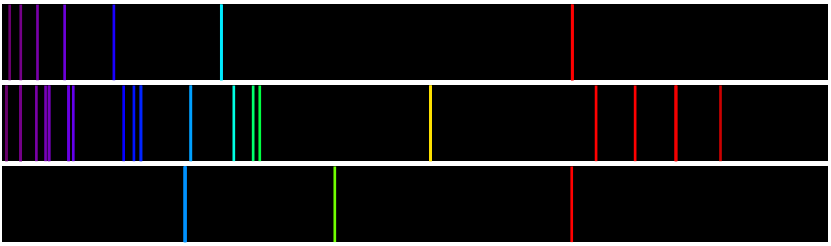
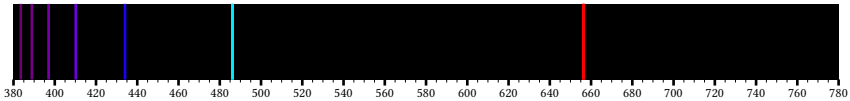


Abb. 6: Paket `pgf-spectra`: Spektren eines Wasserstoffatoms, eines neutralen Heliumatoms und eines einfach ionisierten Heliumatoms

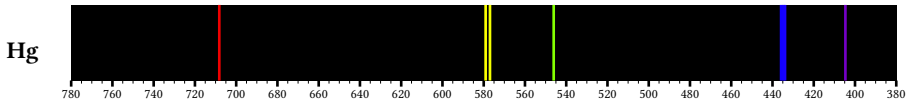


Auf traditionellen Spektraltafeln wird die Wellenlänge von links nach rechts absteigend dargestellt. Dies kann mit Hilfe der Optionen `begin` und `end` erreicht werden. Globale Optionen können mit dem Befehl `\pgfspectraStyle` festgelegt werden. Für die folgenden Darstellungen wählen wir:

```
\pgfspectraStyle[axis,axis ticks=3,axis color=white,axis font color=black,begin
 ↪=780,end=380]
```

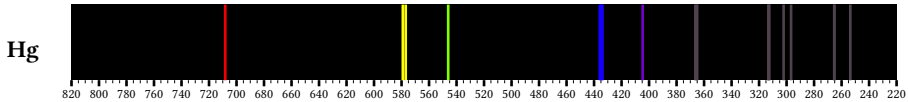
Die Option `label` setzt das chemische Symbol des Elements vor das Spektrum.

```
\pgfspectra[element=Hg,label]
```



Daten zu infraroten und ultravioletten Linien sind ebenfalls vorhanden und können durch geeignete Wahl der Spektrumsgrenzen mit den Optionen `begin` und `end` visualisiert werden.

```
\pgfspectra[element=Hg,label,begin=820,end=220]
```

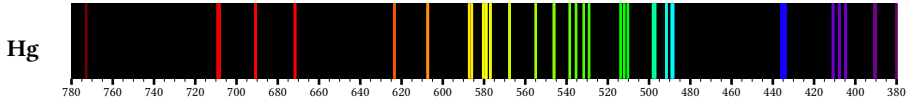


Die Ergebnisse auf Grundlage des paketeigenen Datensatzes sind allerdings nicht immer völlig befriedigend. So weist das Spektrum einer Quecksilberdampflampe eine schwache, aber unter geeigneten Bedingungen deutlich erkennbare türkisblaue Linie mit einer Wellenlänge von 492 nm auf, die in klassischen Spektraltafeln auch vermerkt ist. In der obigen Abbildung fehlt diese allerdings.

Nun bietet das Paket neben dem Standarddatensatz NIST (benannt nach der Datenquelle »National Institute of Standards and Technology«) noch einen zweiten, umfangreicheren Datensatz LSE (»Line Spectra of the Elements« aus dem *CRC Handbook of Chemistry and Physics*), der mit einer entsprechenden Paketoption geladen werden kann.

```
\usepackage[LSE]{pgf-spectra}
```

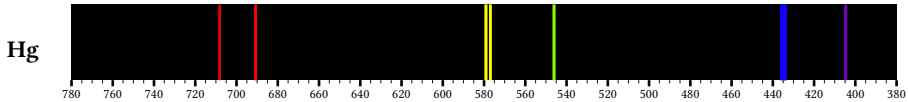
```
\pgfspectra[element=Hg,label]
```



Hiermit taucht die fehlende 492-nm-Linie zwar im Spektrum auf, aber neben dieser auch so viele weitere, dass das Quecksilberspektrum bis zur Unkenntlichkeit entstellt ist. Um weniger intensive Linien auszublenden, definiert das Paket die Option `Imin`, die die minimale relative Intensität der dargestellten Linien festlegt. Für klassische Emissionsspektren empfiehlt die Paketanleitung den Wert `0.05`.

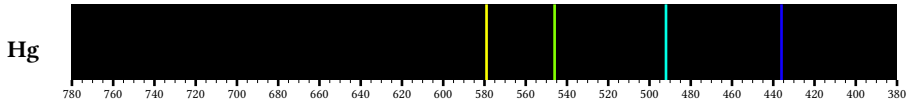
```
\usepackage[LSE]{pgf-spectra}
```

```
\pgfspectra[element=Hg,Imin=0.05,label]
```



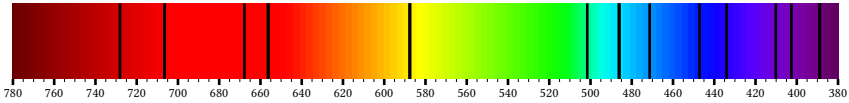
Die türkisblaue Linie ist nun wieder verschwunden, dafür sind rote Linien verblieben, die in der Realität schwächer wahrnehmbar sind als jene. Für solche Fälle ist die manuelle Angabe der Wellenlängen vorzuziehen, beispielsweise:

```
\pgfspectra[lines={436,492,546,579},label,label before text={Hg}]
```



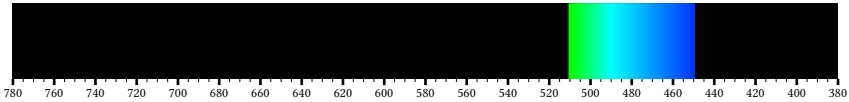
Absorptionsspektren können mit der Option `absorption` gewählt werden. Das folgende Beispiel stellt ein typisches Spektrum eines Sterns dar.

```
\pgfspectra[element={H,He},absorption,Imin=0.05]
```



`pgf-spectra` ermöglicht auch die Darstellung von kontinuierlichen Spektren, wie des folgenden einer blauen Leuchtdiode.

```
\pgfspectra[lines={450 to 510}]
```



Zahlreiche weitere Möglichkeiten des Pakets sind der Anleitung zu entnehmen.

Eng verbunden mit der Untersuchung von Spektrallinien ist die Analyse von Energieniveauschemata von Atomen. Hier würde man sich nun ein Paket wünschen, mit dem ein Energieniveauschema für ein beliebiges Element ebenso einfach erzeugbar ist wie ein Emissionsspektrum mit `pgf-spectra`. Ein solches Paket existiert nicht und kann wohl auch nicht existieren. Mit Ausnahme von Wasserstoff sind die quantenmechanischen Energiezustände der Atomhülle derart komplex, dass sie sich – zumal im Schulunterricht – nur angemessen vereinfacht visualisieren lassen. Die vorgenommene Vereinfachung ist aber immer eine Einzelfallentscheidung, die insbesondere vom untersuchten Emissions- oder Absorptionsphänomen

abhängt, und daher kaum von einem Algorithmus geleistet werden kann. Wer ein Energieniveauschema benötigt, wird es also selber anfertigen müssen.

Wir sehen uns hier zwei Beispiele zur Umsetzung mit Ti*k*Z an (Abb. 7). Das erste zeigt die Energieniveaus des Wasserstoffatoms, für die die Formel $E_n = -13{,}60\,\mathrm{eV} \cdot 1/n^2$ gilt, das zweite ein Energieniveauschema, mit dem sich die charakteristischen Röntgenlinien von Nickel berechnen lassen.[3]

```
\usepackage{tikz}
\usepackage{siunitx}
\usetikzlibrary{arrows.meta}
\sisetup{locale=DE}
```

```
\begin{tikzpicture}[font=\small, scale=0.6]
\draw[semithick, -Stealth] (0,-14)--(0,1) node[right] {$E$ in eV};
\draw[semithick] (0,0) node[left] {0} --(5,0);
\foreach \n in {1,...,4} \draw (0,-13.6/\n^2) node[left] {\pgfmathparse{-13.6/\n
 ↪^2}\num[round-mode=places]{\pgfmathresult}} --(5,-13.6/\n^2);
\foreach \n in {5,...,8} \draw (0,-13.6/\n^2)--(5,-13.6/\n^2);
\end{tikzpicture}
\hspace{2em}
\begin{tikzpicture}[font=\small]
\draw[semithick, -Stealth] (0,0)--(0,8) node[at end, right] {$E$ in keV};
\draw[semithick] (0,0) node[left] {0} --(5.5,0);
\draw[semithick] (0,5) node[left] {7,48} --(5.5,5);
\draw[semithick] (0,6.5) node[left] {8,28} --(5.5,6.5);
\draw[semithick] (0,7.2) node[left] {8,34} --(5.5,7.2);
\draw[->] (1,5)--(1,0) node[midway, left] {K\textsubscript{α}};
\draw[->] (2,6.5)--(2,0) node[midway, left] {K\textsubscript{β}};
\draw[->] (3,7.2)--(3,0) node[midway, left] {K\textsubscript{γ}};
\draw[->] (4,6.5)--(4,5) node[midway, left] {L\textsubscript{α}};
\draw[->] (5,7.2)--(5,5) node[midway, left] {L\textsubscript{β}};
\end{tikzpicture}
```

Aus dem Bereich der Atomphysik soll schließlich noch das Paket `tikzorbital` Erwähnung finden, mit dem sich Elektronenorbitale in Atomen und Molekülen veranschaulichen lassen.

[3] Daten stark vereinfacht nach der »X-ray Transition Energies Database«, https://physics.nist.gov/PhysRefData/XrayTrans/Html/search.html

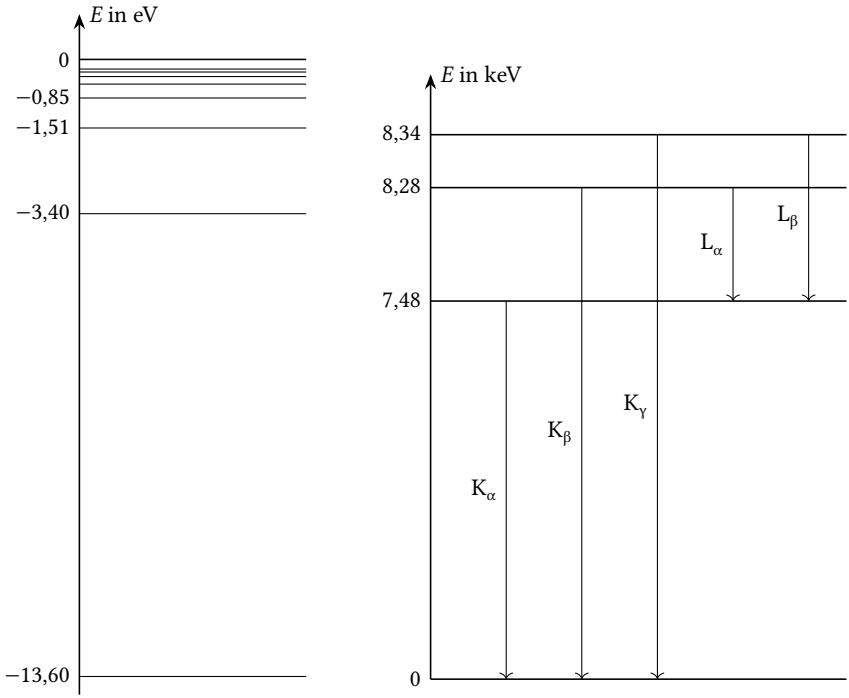


Abb. 7: Energieniveauschemata von Wasserstoff und Nickel, gezeichnet mit Ti*k*Z

Nuklide und Zerfallsreihen

Zur Darstellung von Nukliden empfiehlt sich der Befehl `\isotope` aus dem `chemmacros`-Paket. Er erwartet ein Argument, das die Nukleonenzahl und – durch ein Komma getrennt – das Elementsymbol enthält. Fehlt die Nukleonenzahl, so wird das jeweilige Hauptisotop ausgegeben.

```
\usepackage{chemmacros}
```

```
\isotope{U}\\
\isotope{234,U}\\
\isotope{Ra}\\
\isotope{C}\\
\isotope{14,C}
```

$^{238}_{92}\mathrm{U}$
$^{234}_{92}\mathrm{U}$
$^{226}_{88}\mathrm{Ra}$
$^{12}_{6}\mathrm{C}$
$^{14}_{6}\mathrm{C}$

Durch Verwendung der Sternversion des Befehls wird die Ordnungszahl unterdrückt.

```
\usepackage{chemmacros}
```

```
\isotope*{Rn}\\
\isotope*{220,Rn}\\
\isotope*{Po}\\
```

^{222}Rn
^{220}Rn
^{210}Po

Eine weitere alternative Schreibweise erhält man mit der Option `format=side`. Diese sorgt dafür, dass die Nukleonenzahl hinter das Elementsymbol gestellt wird. Das Trennzeichen wird mit der Option `side-connect` festgelegt, wobei der Standardwert ein Bindestrich ist.

```
\usepackage{chemmacros}
\chemsetup[isotopes]{format=side}
```

```
\isotope{Pb}\\
\isotope{3,H}\\
\chemsetup[isotopes]{side-connect=\,}
\isotope{207,Pb}\\
\isotope{2,H}
```

Pb-208
H-3
Pb 207
H 2

Ein Paket, das – ähnlich wie `pgf-PeriodicTable` für das Periodensystem – beliebige Ausschnitte aus der in der Kernphysik häufig benutzten Nuklidkarte ausgeben kann, existiert leider nicht.

Zur Darstellung von Zerfallsreihen kann man beispielsweise in den Mathematikmodus wechseln und dort den Befehl `\rightarrow` zwischen die einzelnen Nuklide setzen.

```
\usepackage{chemmacros}
```

```
\[\isotope{U}\rightarrow\isotope{234,Th}\rightarrow\isotope{234,Pa}
\rightarrow\isotope{234,U}\rightarrow\isotope{230,Th}\]
```

$$^{238}_{92}\mathrm{U} \rightarrow {}^{234}_{90}\mathrm{Th} \rightarrow {}^{234}_{91}\mathrm{Pa} \rightarrow {}^{234}_{92}\mathrm{U} \rightarrow {}^{230}_{90}\mathrm{Th}$$

Wenn für jeden Zerfall die Zerfallsart angegeben werden soll, kann diese mit Hilfe des Befehls `\stackrel` über den Pfeil geschrieben werden. Dabei sind α und β als

Argumente an den Befehl `\text` zu übergeben, um die Kursivstellung im Mathematikmodus zu vermeiden. Dieser Befehl stammt aus dem Paket `amsmath`, das von `chemmacros` bereits geladen wird.

```
\usepackage{libertinus-otf}
\usepackage{chemmacros}
```

```
\[\isotope{U}\stackrel{\text{α}}{\rightarrow}\isotope{234,Th}
\stackrel{\text{β}}{\rightarrow}\isotope{234,Pa}
\stackrel{\text{β}}{\rightarrow}\isotope{234,U}
\stackrel{\text{α}}{\rightarrow}\isotope{230,Th}\]
```

$$ {}^{238}_{92}\mathrm{U} \overset{\alpha}{\rightarrow} {}^{234}_{90}\mathrm{Th} \overset{\beta}{\rightarrow} {}^{234}_{91}\mathrm{Pa} \overset{\beta}{\rightarrow} {}^{234}_{92}\mathrm{U} \overset{\alpha}{\rightarrow} {}^{230}_{90}\mathrm{Th} $$

Bei Zerfallsreihen mit Verzweigungen bietet sich die Verwendung von Ti*k*Z an, das durch Laden von `chemmacros` bereits zur Verfügung steht. Wir definieren zunächst für jedes Nuklid einen Knoten und verwenden dabei die Ti*k*Z-Bibliothek `positioning`, um die Knoten bequem relativ zueinander positionieren zu können. Dann verbinden wir die Knoten durch beschriftete Pfeile.

```
\usepackage{chemmacros}
\usetikzlibrary{positioning}
```

```
\begin{tikzpicture}[node distance=4mm]
\node (N1) {\isotope{220,Rn}};
\node[right=of N1] (N2) {\isotope{216,Po}};
\node[right=of N2] (N3) {\isotope{212,Pb}};
\node[right=of N3] (N4) {\isotope{212,Bi}};
\node[above right=of N4] (N5) {\isotope{208,Tl}};
\node[below right=of N4] (N6) {\isotope{212,Po}};
\node[above right=of N6] (N7) {\isotope{Pb}};
\tikzset{font=\footnotesize}
\draw[->] (N1)-- node[above] {α} (N2);
\draw[->] (N2)-- node[above] {α} (N3);
\draw[->] (N3)-- node[above] {β} (N4);
\draw[->] (N4)-- node[above left] {α} (N5);
\draw[->] (N4)-- node[below left] {β} (N6);
\draw[->] (N5)-- node[above right] {β} (N7);
\draw[->] (N6)-- node[below right] {α} (N7);
\end{tikzpicture}
```

$$
\begin{tikzcd}
 & & & {}^{208}_{81}\mathrm{Tl} \arrow[dr, "\beta"] & \\
{}^{220}_{86}\mathrm{Rn} \arrow[r, "\alpha"] & {}^{216}_{84}\mathrm{Po} \arrow[r, "\alpha"] & {}^{212}_{82}\mathrm{Pb} \arrow[r, "\beta"] & {}^{212}_{83}\mathrm{Bi} \arrow[ur, "\alpha"] \arrow[dr, "\beta"'] & {}^{208}_{82}\mathrm{Pb} \\
 & & & {}^{212}_{84}\mathrm{Po} \arrow[ur, "\alpha"'] &
\end{tikzcd}
$$

Damit endet die Reihe. Für vielfältige Unterstützung danke ich Herbert Voß.

Literatur

[1] Matthias Kremer, Ulrich Bee: »Mehr Transparenz bei den Elementen mit dem PSE^3!, Ein neues Periodensystem für den Chemieunterricht«, *MNU*, *70.1* (2017), 36–42, `https://asset.klett.de/assets/2a816e3e/MNU_1_2017_36_42.pdf`.

[2] Moritz Nadler: ISO-31-konformer Formelsatz in LaTeX, Version 1.0, 2018, `https://www.moritz-nadler.de/formelsatz.pdf`.

[3] Herbert Voß: »Die Eingabe von Sonderzeichen«, *DTK*, *32.2* (2020), 6–17.

Variable Fonts

Dr. Alexander Willand

Es gibt ein neues Spielzeug bei den Fonts: variable Fonts. Erfreulicherweise hat das LaTeX3-Team die notwendigen Voraussetzungen geschaffen, so dass man variable Fonts unproblematisch mit LaTeX benutzen kann. Zuerst wenden wir uns der Frage zu, was variable Fonts sind, um dann ihre Anwendung vorzuführen.

Viele Jahre lang war »RIBBI« der Standard bei digitalen Fonts. RIBBI steht für »Regular, Italic, Bold, Bold Italic«, also die verbreitetsten Schriftschnitte. Daneben gab es schon immer seit Erfindung des Bleisatzes dünnere und dickere, breitere oder schmalere Schriftsschnitte, heute häufig firmierend zum Beispiel unter »Thin«, »Light«, »Condensed«, »Extra Bold«, »Black«.

Jeder Schriftschnitt lag bisher in einer eigenen Datei, was zu so schönen Dateinamen wie `RobotoCondensed-LightItalic.ttf` führt, der für die kursive Variante des leichteren Schriftschnitts der Roboto steht, oder zu `SourceCodePro-SemiboldIt.otf`.

Alle diese Schriftschnitte konnte man bereits unter pdfTeX nutzen, wenn auch mit gewissem Aufwand, aber darum soll es hier nicht gehen.

Für den möglichst schnellen Aufbau von Internetseiten erwies es sich als zeitraubend, für jeden Schriftschnitt eine eigene Datei laden zu müssen. Stattdessen wurden im Rahmen des OpenType-Formats[1] variable Fonts erfunden. Ein solcher enthält einen besonderen Schriftschnitt, dem der Anwender über die Textsatzsoftware die gewünschte Gestalt verleihen kann.[2]

Kritische Geister werden anmerken, dass diese umfassenden Dateien hunderte von Kilobyte oder sogar mehr als ein Megabyte schwer wurden, was die Ladezeiten wieder verlängerte. Die Lösung dafür bestand darin, dass man die auf dem Server der Website hinterlegte Schrift spezifisch für dieses Webangebot auf diejenigen Features abmagerte, die man gerade brauchte.[3]

Die variablen Fonts, mit denen wir uns hier beschäftigen, halten sich innerhalb einer Schriftgattung. Es wird also nicht aus einer Schrift mit Serifen eine serifenlose und umgekehrt. Wir gehen auch nicht auf eine ähnliche Technik namens Multiple Master Fonts ein.

Variable Fonts haben eine oder mehrere »Gestaltungsachsen«, am häufigsten sieht man eine Achse für die Schriftstärke. Verbreitet ist auch eine Achse für die Schriftweite und für die Neigung, aber es gibt noch viele mehr.[4] Wir kommen darauf zurück.

Eine seltener angebotene Achse regelt die sog. optische Größe einer Schrift. Es liegt auf der Hand, dass eine Überschrift schon aufgrund ihrer Größe viel besser lesbar ist als eine Fußnote. Man kann die Lesbarkeit von kleinen Buchstaben verbessern, indem man ihre Laufweite vergrößert, die Buchstabenformen und die Strichstärke betont. Große Schriften können dagegen enger laufen mit schmaleren Buchstaben, die feiner gezeichnet sind. Die Computer Modern von Donald Knuth verfügt auch über optische Größen und TEX, sowie die darauf beruhenden Programme (LATEX etc.), wählen automatisch die richtige optische Größe aus,[5] sofern die Schrift dies anbietet, was aber bislang selten der Fall war.

---

[1] OpenType ist ein Format für digitale Schriften, das aus dem früheren TrueType hervorging. Neu ist daran, dass man ein Zeichen nicht nur über den zugehörigen Unicode-Wert ansprechen kann, sondern auch über spezielle Features (entnommen `https://www.typografie.info/3/artikel.htm/wissen/mythos-opentype-r270/`).

[2] Allgemein zu variablen Fonts mit Anwendungsbeispielen für Websites siehe `https://www.typografie.info/3/artikel.htm/wissen/was-sind-variable-fonts-r455/` In der TUGboat, Volume 45, S. 12 hat *Peter S. Baker* einen Artikel unter dem Titel »Variable fonts in LuaTEX, with an introduction to the Junicode VF and Elstob fonts« veröffentlicht, der sich mit variablen Fonts vor allem anhand der vom Autor herausgegebenen Fonts Junicode VF und Elstob beschäftigt.

[3] Siehe dazu `https://stackoverflow.com/a/76095573` und `https://www.stefanjudis.com/notes/glyphhanger-a-tool-subset-and-optimize-fonts/` und `https://www.stefanjudis.com/notes/slice-an-app-to-remove-variable-font-axes/` .

[4] Achsen-Beschreibungen: `https://fonts.google.com/variablefonts?vfaxis=opsz#axis-definitions`

[5] Peter S. Baker a. a. O., siehe oben Fussnote 2.

Woher bekommt man eine variable Schrift, möglichst mit einer Achse für optische Größen? Fonts mit einer Achse für optische Größen findet man für Lizenzgebühren natürlich bei Schriftverlagen, aber auch bei Google unter freien Lizenzen: Unter `https://fonts.google.com/variablefonts` gibt es sogar ein Menü, mit dem man die Auswahl auf Schriften mit »opsz« (»optical size«) begrenzen kann.

Mir hat die Schrift Roboto Flex gefallen, die 13 Gestaltungsachsen anbietet. Sie steht auch unter einer freien Lizenz[6], so dass man sie einfach herunterladen und benutzen darf.

Auf die notwendigen Schritte, wie man das heruntergeladene Archiv mit der Schrift entpackt und sie so installiert, das die LaTeX-Installation sie findet, gehe ich hier nicht ein. Klappt schon, oder? Bei dieser Gelegenheit schlage ich vor, die Datei umzubenennen. Ursprünglich lautet der Name
`RobotoFlex-VariableFont_GRAD,XOPQ,XTRA,YOPQ,YTAS,YTDE,YTFI,YTLC,YTUC,opsz,slnt,wdth,wght.ttf`

Das sollten wir ändern: `RobotoFlexVariable.ttf` hört sich brauchbar an.

Experimentieren wir zunächst mit der Schriftstärke. Im folgenden Beispiel beginnen wir die Schriftstärke bei 800 und mindern pro Buchstaben um 50 und gleichzeitig auch die Größe (40 auf 36 auf 32 Punkte, dann jeweils um zwei Punkte). Grau ist alle Theorie, schwarz sehen wir erst in der Praxis:

Z u g d u r c h f a h r t

Für jeden Buchstaben ist ein Befehl mit allen Anweisungen einzugeben, derjenige für das »t« könnte lauten:

```
\itshape\fontsize{14}{10}\selectfont{}
\fontspec{RobotoFlexVariable.ttf}[Weight=150, Slant=-10] t
```

Genauso gut hätte man schreiben können:

```
\itshape\fontsize{14}{10}\selectfont{}
\fontspec{RobotoFlexVariable.ttf}[
RawAxis = {wght=150,slnt=-10} ] t
```

Die Gestaltungsachsen Schriftstärke (»Weight«), Schriftweite (»Width«), Neigung (»Slant«) und optische Größe (»OpticalSize«) lassen sich sowohl über diese Begriffe auswählen, als auch über die Abkürzung für die jeweilige Achse (`wght`, `wdth`,

[6] SIL Open Font License, Version 1.1

`slnt`, `opsz`). Alle weiteren Gestaltungsachsen sind nur über die Abkürzungen zu erreichen.[7]

Kommen wir auf die optischen Größen zurück. Am Beispiel der RobotoFlexVariable probieren wir aus, wie sich die optische Größe einer Schrift auswirkt, indem wir fast den gleichen Text (wer findet den Unterschied?) in der gleichen Schriftgröße, aber in verschiedenen optischen Größen schreiben. Der Unterschied ist in Abbildung 1 deutlich zu sehen. Schön vorgeführt wird dieser Unterschied auch im genannten Artikel von Baker in der TUGboat.

Code zur Abbildung 1:

```
\fontspec{RobotoFlexVariable.ttf}[RawAxis = {opsz = 8}]
Dies ist ein Satz mit Buchstaben in der üblichen Größe der \TeX{}nischen
Komödie in der variablen Schrift RobotoFlex mit der optischen Größe 8.\\
\fontspec{RobotoFlexVariable.ttf}[RawAxis = {opsz = 144}]
Dies ist ein Satz mit Buchstaben in der üblichen Größe der \TeX{}nischen
Komödie in der variablen Schrift RobotoFlex mit der optischen Größe
144. Hammermäßiger Unterschied, oder?!
```

Wie erfährt man, über welche Eigenschaften und Fähigkeiten eine variable Schrift verfügt? Auskunft erteilt das Programm `otfinfo`, das, soweit ich sehe, sowohl mit TEX Live als auch mit MiKTEX[8] mitgeliefert wird. Der Befehl

```
otfinfo --help
```

auf der Kommandozeile zeigt, mit welchen Optionen (das ‘`--help`‘ ist eine Option) Informationen über eine Schrift abgefragt werden können.

Der Befehl

```
otfinfo --variable RobotoFlexVariable.ttf
```

fragt Informationen zu den variablen Eigenschaften ab. Im Ergebnis werden 13 Achsen und 19 benannte Schnitte abgedruckt, letztere werden von otfinfo als »Instance« bezeichnet. Wir drucken einen Ausschnitt aus den Informationen ab:

```
Axis 0:                 opsz
Axis 0 name:            Optical Size
```

[7] The fontspec package, Font selection for XELATEX and LuaLATEX, Abschnitt 7: »Variable Fonts«.
[8] https://github.com/MiKTeX/miktex/issues/457

Dies ist ein Satz mit Buchstaben in der üblichen Größe der TEXnischen Komödie in der variablen Schrift RobotoFlex mit der optischen Größe 8.
Dies ist ein Satz mit Buchstaben in der üblichen Größe der TEXnischen Komödie in der variablen Schrift RobotoFlex mit der optischen Größe 144. Hammermäßiger Unterschied, oder?!

Abb. 1: Vergleich unterschiedlicher optischer Größen

```
Axis 0 range:           8 144
Axis 0 default:         14
Axis 1:                 wght
Axis 1 name:            Weight
Axis 1 range:           100 1000
Axis 1 default:         400
Axis 2:                 GRAD
Axis 2 name:            Grade
Axis 2 range:           -200 150
Axis 2 default:         0 …

Instance 3 name:        Regular
Instance 3 position:    14 400 0 …
```

Wir sehen, dass die Gestaltungsachse Weight = Schriftstärke Werte von 100 bis 1000 annehmen kann. »Instance 3« hat den Namen »Regular« und weist den Achsen 0 bis 12 jeweils ihre Werte zu, um zu einem Schriftschnitt zu kommen, den die Gestalter der Schrift als die normale Grundschrift ansehen. Wir können erkennen, dass für die normale Schrift die optische Größe bei 14 angesetzt wird und die Schriftstärke bei 400. Mir gefällt der Grauwert einer Seite bei einer Schriftstärke von 300 allerdings deutlich besser. Ein kurzes, mit LuaLaTeX zu kompilierendes Beispiel:

```
\DocumentMetadata{lang=de}
\documentclass[parskip=half, ngerman, fontsize=12pt,]{scrartcl}
\usepackage{babel, fontspec}

\setsansfont{RobotoFlexVariable.ttf}[
Renderer = HarfBuzz,
Numbers= Proportional,
ItalicFont= *,
BoldFont = *,
BoldItalicFont = *,
ItalicFeatures = {  Slant = -10, },
BoldFeatures = {  Weight = 600,  Width = 80,  },
BoldItalicFeatures = {  Slant = -10,  Weight = 600,},
]
\renewcommand{\familydefault}{\sfdefault}

\title{RobotoFlexVariable Experimente}

\newcommand{\blindsatz}{Lorem ipsum dolor sit amet, consectetuer
  adipiscing elit.} \date{}

\begin{document}
```

```
\begin{minipage}{1.0\linewidth}
\maketitle{}
\raggedbottom
1234567890

{\addfontfeatures{Weight = 300,}
    \blindsatz (Weight = 300)}

\blindsatz{}\footnote{\blindsatz{}}(Weight = 400)

\emph{\blindsatz{} 1234567890}

\textbf{\blindsatz{}} (Weight = 600)

\textbf{\emph{\blindsatz{}}}
\end{minipage}
\end{document}
```

Das Ergebnis ist in Abbildung 2 leicht kleiner gesetzt abgedruckt. Die Minipage habe ich eingefügt, um die Ausgabe besser abdrucken zu können.[9] Nachträglich eingefügt habe noch die Zeile mit `\addfontfeature`, damit man die leichtere Variante noch vergleichen kann.

Kommen wir nun auf die optischen Größen zurück. Die RobotoFlexVariable nimmt für die optische Größe einen Grundwert von 14 an. Man kann jedoch jeder verwendete Schriftgröße und auch Bereichen ab einer Minimumgröße bis zu einer Maximalgröße eine spezifische optische Größe zuordnen.

Die Anleitung zu »fontspec« enthält einen schönen Trick: Um zu sehen, welche optische Größe tatsächlich verwendet wird, färbt man die damit verbundene Schrift

[9] Und genau genommen ist auch `\maketitle` ersetzt worden durch `{\textbf{RobotoFlexVariable Experimente}}`, weil natürlich der Titel dieses Artikels »Variable Fonts« lautet.

RobotoFlexVariable Experimente

1234567890
Lorem ipsum dolor sit amet, consectetuer adipiscing elit.(Weight = 300)
Lorem ipsum dolor sit amet, consectetuer adipiscing elit.[a] (Weight = 400)
Lorem ipsum dolor sit amet, consectetuer adipiscing elit. 1234567890
Lorem ipsum dolor sit amet, consectetuer adipiscing elit. (Weight = 600)
Lorem ipsum dolor sit amet, consectetuer adipiscing elit.

[a] Lorem ipsum dolor sit amet, consectetuer adipiscing elit.

Abb. 2: Ausgabe des fast minimalen Beispiels mit unterschiedlichen Schriftstärken

RobotoFlexVariable mit unterschiedlichen optischen Größen

1234567890
Lorem ipsum dolor sit amet, consectetuer adipiscing elit.[a]
Lorem ipsum dolor sit amet, consectetuer adipiscing elit. 1234567890
Lorem ipsum dolor sit amet, consectetuer adipiscing elit.
Lorem ipsum dolor sit amet, consectetuer adipiscing elit.

[a] Lorem ipsum dolor sit amet, consectetuer adipiscing elit.

Abb. 3: Ausgabe mit geänderten optischen Größen

ein. Wir ergänzen die Optionen nach `\setsansfont{RobotoflexVariable.ttf}[` … um Anweisungen für Größen. Allerdings muss man für jede Variante (normale Schrift, fett, kursiv…) diese Anweisung aufnehmen:

```
SizeFeatures = {
{Size = -10, OpticalSize = 16, Color=FF000099},%rot
{Size=10-13, OpticalSize=36, Color=0000FF99},%blau
{Size=13-, OpticalSize=72, Color=00BB3399}},%grün
```

Abbildung 3 zeigt das Ergebnis.

Man kann hier mit einer gewissen Sorgfalt und Spieltrieb sehr schön Schriftgröße, Schriftstärke und optische Größen aufeinander abstimmen:

Mir erscheint die etwas leichtere und etwas schmäler laufende Auslegung (Width = 90, statt = 100) als besser lesbar. Das schöne ist, wem es anders besser gefällt, der kann einfach an den Gestaltungsachsen herumprobieren, bis ihm oder ihr das Ergebnis mehr zusagt. Insbesondere die Kombination der Achsen für die Schriftstärke mit derjenigen für optische Größen halte ich für einen erheblichen Vorteil variabler Fonts.

Man kann natürlich auch mit allen 13 Achsen von RobotoFlexVariable die Schrift in abenteuerlicher Weise entstellen. Was würde zum Beispiel passieren, wenn wir die Achsen XOPQ, YOPQ und XTRA auf 30, 30 und 323 setzen?

RobotoFlexVariable

Man erhält eine deutlich anders anmutende Schrift. Und dabei haben wir bisher nur vier von 13 Gestaltungsachsen verwendet. Was passiert eigentlich, wenn wir YTUC von 712 auf 528 setzen, sodann noch die Laufweite

drastisch nach oben korrigieren, die Werte für YTAS (»parametric Ascender Height«) und YTDE (»parametric Descender Depth«) auf ihren Maxmial- bzw. Minimalwert setzen und dann den Zauberstab noch über »GRAD« schwingen? Leider konnte ich die `listings`-Umgebung nicht davon überzeugen, den Text darin mit RobotoFlexVariable auszudrucken, aber ich hoffe, die Leserin und den Leser vom Wert der variablen Schriften überzeugt zu haben. Der Code für die Überschrift und die ersten Zeilen dieses letzten Absatzes lautet übrigens:

```
\addfontfeature{BoldFeatures= {RawAxis = {XOPQ = 30, YOPQ = 30, XTRA
      = 323}},
  UprightFeatures = {RawAxis = {wght=350, XOPQ = 30, YOPQ = 30, XTRA=
      323, GRAD = 150}}} \LARGE\textbf{RobotoFlexVariable}
```

und dann habe ich jeweils `\addfontfeature =...` hinzugefügt. Geht doch!

Aufgeschnappt

Luzia Dietsche

Immer mal wieder schnappe ich beim Surfen im Internet oder beim Lesen verschiedener Mailinglisten Hinweise auf, die vielleicht auch andere interessieren.

Ottmar-Mergenthaler-Museum

Ein Museum im Geburtshaus Mergenthalers erinnert an den genialen Erfinder aus dem Taubertal.

> *Unzählige Zeitschriften, Bücher in Millionenauflage, dicke Kataloge: Das gedruckte Wort ist heute überall. Den Grundstein dafür hat ein Erfinder aus Bad Mergentheim gelegt. Ottmar Mergenthaler, der gelernte Uhrmacher, wanderte 1872 in die USA aus und arbeitete in der Instrumentenbaufirma seines Vetters in Baltimore. 1886 konstruierte er die Linotype: die erste brauchbare Setzmaschine, die ganze Zeilen („line of types") statt einzelner Buchstaben ausgoss.*

Informationen zu Öffnungszeiten und Anschrift findet man unter `https://visit.bad-mergentheim.de/de/kultur-schloss-genuss/ottmar-mergenthaler-museum/`.

Sehr anschaulich wird im folgenden Artikel über das Museum berichtet: `https://realdougwilson.com/writing/the-box-at-the-museum`

TUGIndia

Die indische TEX Users Group wurde am 7. September 2024 nach 32 Jahren wiederbelebt. Der aktuelle Vorstand besteht aus folgenden Personen:

- Präsident: E. Krishnan, e.krshnan@gmail.com
- Vizepräsident: Rajeesh, KV rajeesh@sayahna.org
- Schriftführer: Niranjan, niranjan.t@stmdocs.in
- Gemeinsamer Sekretär: Rahul Krishnan S, rahulkrishnan@stmdocs.in
- Rechtlicher Beirat: Aravind Ranjendran

Als erstes Projekt von TUGIndia freuen wir uns, folgendes zu veröffentlichen: ein LATEX-Paket *udiss*, das der Unterstützung von Studierenden beim Setzen ihrer Dissertationen dient.

Vorträge der TUG'24 in Prag

Die Vorträge der Tagung TUG'24 in Prag sind inzwischen alle dank Norbert Preining auf dem YouTube-Kanal der TEX Users Group verfügbar: https://tug.org/l/tug24-video

ConTEXt kurz notiert[1]

Henning Hraban Ramm

In der ConTEXt-Welt ist vieles in Bewegung, das eine Erwähnung wert ist, aber keinen ganzen Artikel begründet. (Stand: 4. 11. 2024)

ConTEXt Meeting 2024

Für das diesjährige Treffen hat Ton Otten (Hans Hagens Geschäftspartner bei Pragma ADE) uns im ehemaligen Wasserturm von Lutten bei Hardenberg untergebracht, den der heutige Besitzer zu einer gemütlichen Unterkunft ausgebaut hat. Durch die Nähe zum Pragma-Hauptquartier war auch Tons Familie eingespannt: seine Frau Annelies, eine Filzkünstlerin, bot als handwerklichen Teil an, Kunstobjekte oder Laptop-Matten zu filzen, und seine erwachsenen Kinder kamen uns für einen Tag besuchen (Bram kannten wir schon aus Sibřina). Überhaupt war der

[1] Die verwendeten Fotografien stammen von Hraban Ramm und Harald König

Abb. 1: ConTEXt-Entwicklerteam mit Fahrradschülerin und Publikum

Altersdurchschnitt relativ niedrig, weil weitere Student*innen teilnahmen und Ryszard Kubiak eine ukrainische Gastschülerin mitbrachte. Daria hatte schon beim BachoTEX-Treffen ihr Schul-Kunstprojekt über das Innenleben von Büchern vorgestellt und wiederholte ihre gelungene Präsentation für uns. Ein paar von uns hatten dann Spaß, ihr das Fahrradfahren beizubringen.

Eines der Themen, die sich wiederholten, waren Workflows für die Produktion von Büchern: Während Taco Hoekwater den Weg über RTF gewählt hat und einen eigenen Parser für dieses schlecht spezifizierte Microsoft-Format vorgestellt hat, exportiert Keith McKay ePub aus LibreOffice und verwendet die XML-Eingabe von ConTEXt, um das ePub-HTML zu interpretieren. Die Bücher meines Verlags nehmen den Weg von DOCX über ein Python-Skript, das ConTEXt-Markup aus dem enthaltenen XML erzeugt. Pandoc mit seinem ConTEXt-Backend ist eine weitere Möglichkeit, und Massimiliano Farinella verwendet das interne Format von Pandoc in seinem Prosemirror-basierten Editor, der sich je nach Projekt auf verschiedene Ein- und Ausgabeformate konfigurieren lässt, nicht zuletzt ConTEXt und InDesign-XML.

Hans Hagen und Mikael Sundqvist fassten die Entwicklungen des vergangenen Jahres zusammen, über die ich hier schon großenteils berichtet habe. Weitere sind unten genannt.

Die Exkursion führte uns diesmal zum größten Hünengrab der Niederlande und dem zugehörigen Museum, wo u. a. verschiedene frühgeschichtliche Epochen (Stein-/Bronze-/Eisenzeit) durch nachgebaute Häuser dargestellt waren, bis hin zu einem

Abb. 2: Spaß am Gerät

Feld mit alten Getreidesorten, das auch mit historischen Methoden bestellt worden war.

Handbücher

Alle Teilnehmer des ConTEXt Meetings konnten ein gedrucktes Exemplar des neuen Mathe-Handbuchs mit nach Hause nehmen – 288 Seiten, davon 58 Seiten Symbolverzeichnis und 23 Seiten Schriftmuster der Matheschriften. Das PDF in Druck- und Bildschirmversion ist als Teil der LMTX-Distribution frei verfügbar.

Weitere Dokumentations-PDFs wurden erweitert und ergänzt, besonders in der `lowlevel`-Reihe.

Aufrufskripte

Bekanntlich hat das zentrale Skript `mtxrun` zahlreiche Optionen und Zusatzskripte. Neu ist dabei `fixpdf`, mit dem sich PDFs analysieren und auf verschiedene Weise neu schreiben lassen, z. B. unkomprimiert.

An ConTEXt MkII gibt es schon seit vielen Jahren keine Änderungen mehr, aber das alte Skript `texutil` soll jetzt doch noch durch eine Lua-Version ersetzt werden, um die Abhängigkeit von Ruby aus TEXLive zu entfernen.

Abb. 3: Gruppenbild mit Wasserturm

SVG-Export

Beim HTML-Export erzeugt ConTEXt für Formeln MathML. Dieses wird allerdings von Browsern äußerst inkonsistent wiedergegeben, die verwendeten Schriften sind eher zufällig. Um die hohe Qualität des Formelsatzes zu erhalten, ist jetzt auch der Export nach SVG möglich. Dazu muss `context` mit dem Parameter `--images` aufgerufen werden. Der SVG-Export lässt sich mit der Umgebung `\startimage` … `\stopimage` auch für andere Objekte aktivieren.

Schrift-liches

Alte Bitmap-Schriften (`pk`) können »on-the-fly« mit den integrierten `potrace`-Funktionen in OpenType-Outlines umgewandelt werden. (Wenn ich es richtig verstanden habe, geschieht das auch beim Platzieren von PDFs.) Die vorgeführten Ergebnisse sahen zumindest nicht schlechter aus als die Originale und sparen etwas Speicherplatz und Rechenzeit bei der Darstellung.

Einige eher exotische Schriftprobleme werden jetzt umgangen (etwas mit CFF, Weißraum bei Satzzeichen und Mathe-Tabellen in Textschriften).

Rechnen damals und anderswo

Immer für einen historischen Ausblick gut, führte Willi Egger uns diesmal in Rechenmaschinen aus verschiedenen Zeitaltern und Weltgegenden ein. Als handwerklichen Teil durfte jede/r von uns einen Mini-Soroban (japanischer Abakus) zusammensetzen.

Ein wenig moderner, stellte Bruce Horrocks sehr ausführlich die Produktpalette der HP-Taschenrechner vor, mit vielen Exponaten aus seiner Sammlung. Der ConTEXt-Bezug war dabei, dass er ein »Fanzine« für Sammler herausgibt.

Schließlich gab uns Frits Spijkers, der ansonsten für die Übungen im Math4All-Projekt[2] verantwortlich ist, Rechenunterricht mit den Kaktovik-Zahlen, mit denen wir uns schon beim vorigen Treffen beschäftigt hatten.

Miszellen

Der Preis für den umständlichsten Einsatz von ConTEXt geht an Riviera Taylor: Wir bekamen eine Vorführung von Live-Coding mit »Tidal Cycles«, das eigentlich für Musik gedacht ist.

Zurück zu praktischeren Neuigkeiten:

[2] `https://www.math4all.nl`, deckt den niederländischen Mathe-Lehrplan ab.

Der Makro-Parser versteht jetzt auch geschachtelte eckige Klammern, so dass Befehle in `\setup`-Befehlen keine zusätzliche Gruppe (geschweifte Klammern) mehr brauchen.

Es gibt noch mehr Maßeinheiten, weil es so einfach ist, sie zu definieren und weil sie Setups übersichtlicher machen können, z. B. kann `.5\textwidth` auch als `.5tw` angegeben werden. Alle Maßeinheiten sind auch in MetaPost verwendbar. Ähnlich lässt sich auch auf die Parameter der aktuellen Schrift und der aktuellen Umgebung (Spalte, Rahmen) zugreifen. (Nein, die imperialen Einheiten haben wir immer noch nicht abgeschafft.)

Der mehrstufige Absatzumbruch ist längst benutzbar und enthält jetzt auch den vertikalen Zeilenausgleich (`vz`) – wenn man auf registerhaltigen Satz verzichten kann, bekommt man so mit wenig Aufwand gute Umbruch-Ergebnisse (bei einem Roman mit 648 Seiten war mir das neulich eine große Hilfe).

Wolfgang Schuster hat die Interface-XML-Dateien, also die formale Befehlsreferenz, noch einmal durchgesehen und auf den neuesten Stand gebracht.

MetaPost (bzw. MetaFun) hat weitere Operatoren und Funktionen bekommen, auch zur Unterstützung von SVG. Der Quellcode wird von CWeb auf einfaches C umgestellt.

Geplante Änderungen

Als Bereiche, an denen er in nächster Zeit arbeiten wird, nannte Hans Hagen den Mehrspaltensatz (`columns` und `columnsets`), die Platzierung von Gleitobjekten, die Verbesserung von TaggedPDF, die Unterstützung von CSL[3] für Bibliografien und die Darstellung von chemischen Strukturformeln (das alte `PPCHTEX` funktioniert nicht mehr). Ein kleines Modul für die Darstellung von Elektronenschalen (wenn ich das richtig verstanden habe), ist neulich von mehreren Beteiligten auf der Mailingliste entwickelt worden.

Restliche 8-bit-Funktionen und das obsolete LuaJIT werden entfernt (LuaMetaTEX enthält die Entwicklungsversion von Lua 5.5; LuaJIT basiert auf Lua 5.1 und ist beim Einsatz mit TEX eher langsamer).

ConTEXt Meeting 2025

Das nächste ConTEXt Meeting wird vom 23. bis 29. August in Polen stattfinden, und zwar im Feriengelände *Krefta*[4] in der Nähe von Gdańsk. Die Anlage befindet

[3] Citation Style Language, `https://citationstyles.org`
[4] `https://krefta.pl`, Website nur auf polnisch.

sich zwischen zwei Seen und ist mit der von Bachotek vergleichbar, der lokale Organisator ist GUST-Sekretär Ryszard Kubiak.

ASN-Labels mit LaTeX

Uwe Ziegenhagen

»Digitalisierung der Dokumentenablage« ist ein Thema, das auch für mich immer relevanter wird. In diesem Artikel beschreibe ich, wie ich die hilfreichen ASN-Labels mit LaTeX erzeuge.

Worum geht es?

Schon seit einigen Jahren versuche ich – durch sinnvolles digitales Archivieren und Wegwerfen – der Papierlast durch eingehende Dokumente Herr zu werden. Im OpenSource-Bereich erfährt vor allem `paperless-ngx` große Beliebtheit, was der relativ einfachen Installation und dem großen Funktionsumfang geschuldet ist. Die c't hatte vor einiger Zeit eine Anleitung für Docker veröffentlicht (`https://heise.de/-9672632`), der ich weitestgehend gefolgt bin.

`Paperless-ngx` unterstützt dabei auch die Kombination aus digitaler und physischer Ablage. Die Idee dabei ist, nicht mehr dutzende unterschiedlicher Aktenordner zu haben, in denen die physischen Dokumente thematisch und zeitlich abgelegt werden, sondern viel mehr nur noch Aktenordner, in denen die Dokumente nach dem zeitlichen Eingang liegen. Das Chaos beim Suchen einzelner Dokumente soll dadurch verhindert werden, dass jedes Dokument eine ASN (»Archive Serial Number«) erhält, eine fortlaufende ID. Diese wird in Form eines »Aufklebers« auf dem Dokument angebracht, `paperless-ngx` ist dann beim Verarbeiten der Dokumente in der Lage, diese ASN auszuwerten.

Diese ASN-Aufkleber lassen sich über passende Generatoren wie `https://tobiasmaier.info/asn-qr-code-label-generator/` online erzeugen, aber – der geneigte Leser oder die geneigte Leserin wird es bereits ahnen – man kann sie natürlich auch mit LaTeX erzeugen.

Umsetzung

Die Aufgabe unterteilt sich in zwei Schritte: a) baue eine Vorlage für die Aufkleber und b) befülle sie einem fortlaufenden QR-Code sowie der menschenlesbaren Nummer.

Die Vorlage für die Aufkleber

Die Aufkleber erstelle ich mit dem von mir hochgeschätzten `ticket`-Paket. Die Definition der Tickets kommt in eine `filecontents*`-Umgebung, die beim TeXen dann die Datei `ASN.tdf` schreibt. Die `overwrite`-Option sorgt dafür, dass die Ticket-Definition immer wieder neu geschrieben wird, was insbesondere bei der Definition der diversen Längen hilfreich ist.

```
\begin{filecontents*}[overwrite]{ASN.tdf}
\unitlength=1mm
\hoffset=-16mm % Position abhängig vom Drucker
\voffset=-8mm % Position abhängig vom Drucker
\ticketNumbers{4}{12} % vier Spalten und 12 Zeilen pro Blatt
\ticketSize{45.7}{21.2} % Breite und Höhe der Labels in mm
\ticketDistance{2.5}{0} % Abstand der Labels horizontal/vertikal
\end{filecontents*}
```

`\usepackage[ASN,boxed]{ticket}` kommt dann *unter* die `filecontents*`-Umgebung. Damit verhindert man, dass sich LaTeX beschwert, wenn man die Datei zum ersten Mal übersetzt und die `ASN.tdf` noch nicht existiert. Die `boxed`-Option sorgt für die Rahmen um die einzelnen Labels, vor dem finalen Druck auf die entsprechenden Label-Bögen sollte man diese Option daher entfernen.

Erzeugen der Labels

Für das Erzeugen der QR-Codes bietet sich das gleichnamige Paket an, das einfach den Text übergeben bekommt, den es als QR-Code anzeigen soll. In unserem Fall ist das »ASN«, gefolgt von der fünfstelligen Nummer, die links mit Nullen aufgefüllt wurde. Diese aufgefüllten Zahlen zu erzeugen, war meines Erachtens die größte Herausforderung, die ich nur dank `tex.stackexchange.com` bewältigen konnte. An dieser Stelle auch noch einmal Dank an Ulrike Fischer, die den Hinweis zum »expandable padding command« gab. Die Antwort in `https://tex.stackexchange.com/a/250328` hatte die entsprechende Definition. Dies, zusammen mit ein wenig »Schleifenmagie« für das Erzeugen der 48 individuellen Label mit definierbarem Startwert, führte zum finalen Code, der im folgenden Listing abgedruckt ist und auch auf meiner Homepage unter `https://www.uweziegenhagen.de/?p=4715` zur Verfügung steht.

```
\documentclass[a4paper,12pt]{scrartcl}
\usepackage[total={210mm,297mm},top=0mm,left=0mm,bottom=0mm,includefoot]{geometry
 ↪}
\usepackage{qrcode}
\usepackage{forloop}
\usepackage[T1]{fontenc}
```

```
% https://tex.stackexchange.com/questions/716116/generate-sequential-padded-
 ↪barcodes-with-qrcode
\makeatletter
\newcommand{\padnum}[2]{%
  \ifnum#1>1 \ifnum#2<10 0\fi
  \ifnum#1>2 \ifnum#2<100 0\fi
  \ifnum#1>3 \ifnum#2<1000 0\fi
  \ifnum#1>4 \ifnum#2<10000 0\fi
  \ifnum#1>5 \ifnum#2<100000 0\fi
  \ifnum#1>6 \ifnum#2<1000000 0\fi
  \ifnum#1>7 \ifnum#2<10000000 0\fi
  \ifnum#1>8 \ifnum#2<100000000 0\fi
  \ifnum#1>9 \ifnum#2<1000000000 0\fi
  \fi\fi\fi\fi\fi\fi\fi\fi\fi
  \expandafter\@firstofone\expandafter{\number#2}}
\makeatother

\begin{filecontents*}[overwrite]{ASN.tdf}
\unitlength=1mm
\hoffset=-16mm  \voffset=-8mm
\ticketNumbers{4}{12}
\ticketSize{45.7}{21.2} % Breite und Höhe der Labels in mm
\ticketDistance{2.5}{0} % Abstand der Labels
\end{filecontents*}

\usepackage[ASN,boxed]{ticket} % boxed vor dem Druck entfernen
\renewcommand{\ticketdefault}{} % Reset Background
\newcounter{asn} % start value of the labels
\setcounter{asn}{1}

\newcommand{\mylabel}{
\ticket{\hspace*{4mm}\raisebox{9mm}[4mm][2mm]{%
\qrcode[height=1.4cm]{ASN\padnum{5}{\value{asn}}} %
~\texttt{\large ASN\padnum{5}{\value{asn}}}
} \stepcounter{asn}
}}

\begin{document}
\newcounter{x} % für die Schleife
% create 48 labels
\forloop{x}{1}{\value{x} < 49}{\mylabel}
\end{document}
```

Abb. 1: Verkleinerte Version eines Labels-Bogens

Von fremden Bühnen

LaTeX News – Issue 39, June 2024[1]

Frank Mittelbach

Einleitung

Das LaTeX-Projektteam konzentriert sich weiter stark auf das Erzeugen einer automatisch getaggten PDF-Ausgabe – für die Barrierefreiheit und zum besseren Wiederverwenden. Anfang 2024 wurden die Standards ISO PDF/UA-2 und WTPDF (Well-Tagged PDF) veröffentlicht und wir freuen uns, berichten zu können, dass es mittlerweile möglich ist, mit LaTeX Dokumente erstellen zu können, die diese neuen Standards erfüllen.[2] Eine Sammlung an Beispielen für solche Dokumente – von klassischen Texten wie der Bibel bis hin zu aktuellen technischen Artikeln, die auf `https://arXiv.org` eingereicht wurden – finden Sie unter `https://github.com/latex3/tagging-project/discussions/72`.

Im Februar haben Ulrike und Frank den aktuellen Projektstatus im Rahmen des 5th International Workshop zu »Digitization and E-Inclusion in Mathematics and Science 2024« (DEIMS 2024) an der Nihon-Universität, Tokyo, Japan vorgestellt; siehe [2].

Neues vom »LaTeX Tagged PDF«-Projekt

In den letzten LaTeX News [4, S. 97] haben wir eine erste Unterstützung für getaggte Tabellenumgebungen angekündigt. Ein Teil des dafür notwendigen Codes wurde nun aus `latex-lab` in die entsprechenden Pakete (mit Hilfe von Sockets und Plugs) und in den LaTeX-Kernel (für die Teile, die auch für andere Aspekte des Tagging erforderlich sind) umgezogen.

[1] Der Newsletter wurde zuvor im *TUGboat* 45.2 [3] veröffentlicht und von Thomas Demmig übersetzt.

[2] Aktuell befinden wir uns noch in einer Trial/Prototypen-Phase, in der nur eine begrenzte Zahl von Dokumentklassen und Paketen unterstützt werden. Im Laufe der nächsten Releases wollen wir diese Beschränkungen nach und nach abbauen und schließlich die vollständige Funktionalität als Teil der Core Distribution bereitstellen, statt `latex-lab`-Module nutzen zu müssen.

Der Kernel-Code für das Tagging ist in der Datei `lttagging.dtx` implementiert. Momentan enthält er `\UseTaggingSocket`, einen speziellen Befehl zur Ausführung von Sockets, die für Tagging genutzt werden. Damit konnten wir dann auch `\SuspendTagging` und `\ResumeTagging` bereitstellen – ein sehr effizienter Weg zum temporären Deaktivieren des gesamten Tagging-Prozesses. Das ist zum Beispiel für ein Trial-Typesetting erforderlich, da nicht die Trials, sondern nur die abschließend gewählte Version dann Tagging-Strukturen generieren soll. Daher stoppt `tabularx` beispielsweise das Tagging bei seinen Trials, mit denen die korrekte Spaltenbreite ermittelt werden soll, und aktiviert es erst dann wieder, wenn die Tabelle endgültig gesetzt wird.

Mit der Zeit wird `lttagging.dtx` mehr allgemeinen Tagging-Code aufnehmen. Aktuell ist der Code nur als Teil von `source2e.pdf` dokumentiert, aber langfristig werden wir einen eigenen Guide für das Tagging erstellen, der dann auch die Informationen enthalten wird, die momentan auf unterschiedliche Stellen verteilt sind, zum Beispiel in `tagpdf.pdf`.

Wir haben außerdem die Unterstützung für ein paar fehlende Befehle hinzugefügt, die in Leslie Lamports *LaTeX Manual* [1] beschrieben sind: Wenn `phase-III` genutzt wird, wird der Befehl `\marginpar` (abhängig von der PDF-Version) sauber als `Aside`- oder `Note`-Struktur getaggt. In den Standardklassen wird `\maketitle` getaggt, wenn das zusätzliche Testphasen-Modul `title` zum Einsatz kommt.

Das Modul `math` wurde erweitert und enthält nun Optionen, um MathML-Dateien an die Strukturen anzufügen. Erste Tests mit einem PDF Reader und Screen Reader, die solche assoziierten Dateien unterstützen, sind sehr vielversprechend. Beispiele für PDF-Dateien, die mit der neuen Methode getaggt wurden, finden Sie unter `https://github.com/latex3/tagging-project/discussions/72`.

Zudem wurden eine Reihe kleinerer Fehler und Probleme behoben, die über `https://github.com/latex3/tagging-project` gemeldet wurden. Solches Feedback ist sehr wertvoll, daher hoffen wir, auch Sie dort zu sehen und danken Ihnen für jede Art von Beitrag, sei es ein Problem oder ein Post in einem Diskussions-Thread.

Verbesserungen am neuen Markierungsmechanismus

Im Juni 2022 haben wir einen neuen Markierungsmechanismus vorgestellt [4, S. 76], der es ermöglicht, mehrere unabhängige Markierungen zu verwalten. Zudem unterstützt er Top-Marks sauber – etwas, das zuvor mit LaTeX nicht möglich war.

Es gab allerdings eine Einschränkung: Um die Markierungen aus den Seitendaten zu erhalten, mussten diese Daten per `\vsplit` getrennt werden, so dass TeX Split-Marks erzeugen konnte, die der Mechanismus dann verstand. TeX wird allerdings ausgesprochen ungehalten, wenn es innerhalb dieser Daten negativen Glue findet

(zum Beispiel von \vss). Das ist nicht allzu überraschend, weil solcher Glue das Absplitten beliebiger Mengen an Material ermöglichen würde, da er dessen Größe verbirgt. TeX reagiert daher mit einer Fehlermeldung, wenn es solchen Glue in einer \vsplit-Operation findet (und das auch, wenn später ein Glue-Element den unendlichen Glue neutralisiert).

Um das zu berücksichtigen, versuchte der Code 2022, solche Situationen im Vorhinein zu erkennen und dann kein Absplitten durchzuführen, aber dann wurden natürlich auch keine Markierungsinformationen extrahiert.

In diesem Release wurde das Vorgehen geändert und wir führen immer eine \vsplit-Operation durch, um damit immer die richtigen Markierungsdaten zu extrahieren. Es ist zwar nicht möglich, TeX bei unendlich negativem Glue nicht in Rage zu bringen, aber es ist möglich, dies (mehr oder weniger) vor dem User zu verbergen.[3] Mit dem neuen Code wird TeX weder stoppen noch etwas im Terminal ausgeben. Wir können allerdings nicht verhindern, dass ein Fehler in die Logdatei geschrieben wird, aber um klarzustellen, dass dieser Fehler harmlos ist und ignoriert werden kann, haben wir den Code so angepasst, dass die Fehlermeldung, wenn sie ausgegeben wird, folgendes Format hat:

```
! Infinite glue shrinkage found in box being split.
<argument> Infinite shrink error above ignored !
l. ... }
```

Das ist nicht perfekt (insbesondere das etwas unmotivierte ‹argument›), aber man hat als Entwickler nur einen eingeschränkten Handlungsspielraum, wenn die Fehlermeldungen und ihre Texte fest verdrahtet sind.

Warum nun all das? Es gibt zwei Gründe: Wir verlieren keine Markierungen mehr in Randfällen und – vielleicht noch wichtiger – sind wir nun auch in der Lage, Markierungen aus beliebig »verpackten« Daten zuverlässig zu extrahieren, was zuvor überhaupt nicht möglich war. Dies ist notwendig, um beispielsweise erweiterte Markierungen in multicols-Umgebungen zu unterstützen oder sie aus Gleitumgebungen, Randnotizen usw. zu extrahieren.

Details zur Implementierung erhalten Sie, wenn Sie texdoc␣ltmarks-code ausführen oder durch texdoc␣ltmarks-doc (was nur die allgemeinen Konzepte und die Befehlsschnittstellen beschreibt).

[3] Ein Hinweis für Anwender von l3build, die dessen Testfunktionalitäten nutzen: Der neue Mechanismus ändert temporär \interactionmode, was aus Implementierungsgründen in TeX zu zusätzlichen Zeilenumbrüchen in der .log-Datei führt. Statt [1]␣[2] erscheinen diese als einzelne Zeilen. Die Testdateien werden daher dann eventuell solche Unterschiede aufweisen, sobald der Code aktiv ist, und müssen daher gegebenenfalls neu generiert werden.

xtemplate im Format bereitstellen

In den LaTeX News 32 haben wir die Integration einer lange experimentellen Idee in den Kernel beschrieben: Das Paket `xparse`, das als `ltcmd` integriert wurde, so dass seine Funktionalität immer zur Verfügung steht. Mit dieser Version haben wir eine weitere lang existierende Entwicklungsidee in einen stabilen Status gebracht: sogenannte *Templates*.

Im LaTeX-Kontext sind Templates ein Mechanismus zum Abstrahieren verschiedener Elemente eines Dokuments (wie zum Beispiel den Überschriften) auf eine Art und Weise, dass unterschiedliche Implementierungen austauschbar verwendet werden können und Design-Entscheidungen separat, effizient und kontrollierbar implementiert werden können.

Im Gegensatz zu `ltcmd`, das einen Mechanismus bietet, welcher von vielen Dokumentautoren regelmäßig genutzt wird, sind Templates ein spezialisierteres Werkzeug. Wir erwarten daher, dass es im Wesentlichen nur von Paket-Entwicklern benutzt wird, die damit abstrakte und konfigurierbare Objekte (Templates) bereitstellen, die dann in Dokumentklassen zum Einsatz kommen. Die meisten Dokumentautoren werden mit der Template-Entwicklung kaum in Berührung kommen – wir gehen aber davon aus, dass sie die vom Team und anderen Entwicklern entwickelten Templates verstärkt *nutzen* werden (etwa um Konfigurationen zu ändern).

Das Template-System besteht aus drei separaten Ideen:

- Template-*Typ*: Das abstrakte »Ding«, für das wir Templates nutzen, wie zum Beispiel »Überschriften« oder »nummerierte Listen«
- Ein Template: Eine Kombination aus Code und Schlüsselwörtern, die zum Implementieren eines Typs genutzt werden kann. Hier haben wir vielleicht zum Beispiel »standard-LaTeX-sectioning« als ein Template für »Überschriften« oder »KOMA-sectioning« für ein weiteres Template für den gleichen Typ
- Eine oder mehrere *Instanzen*: Ein spezifischer Anwendungsfall eines Templates, bei dem (einige) Schlüssel explizite Werte erhalten. Wir könnten beispielsweise »LaTeX-section«, »LaTeX-subsection« usw. sehen.

Als Teil der Integration des experimentellen `xtemplate`-Pakets in den Kernel hat das Team die bereitgestellten Befehle überarbeitet und zum Teil geändert und erweitert. Der finale Satz besteht nun aus

- `\NewTemplateType`,
- `\DeclareTemplateInterface`,
- `\DeclareTemplateCode`,
- `\DeclareTemplateCopy`,
- `\EditTemplateDefault`,
- `\UseTemplate`,
- `\DeclareInstance`,
- `\DeclareInstanceCopy`,
- `\EditInstance`,
- `\UseInstance`,
- `\IfInstanceExistsTF` und seinen Varianten.

Um bestehende Pakete, die `xtemplate` verwenden, zu unterstützen, haben wir eine aktualisierte Version von `xtemplate` veröffentlicht, die problemlos mit dem neuen Code auf Kernel-Level zusammenarbeitet. Die bestehenden Befehle aus `xtemplate` werden also weiterhin funktionieren, aber wir wünschen uns, dass Programmierer zum oben aufgeführten Satz an Kommandos wechseln.

Neue oder verbesserte Befehle

`doc`: Bereitstellen von `\ProvideDocElement`

Neben `\NewDocElement` und `\RenewDocElement` bieten wir nun auch die Deklaration `\ProvideDocElement` an, die nichts tut, wenn das Element nicht mit `\NewDocElement` deklariert werden kann, weil es schon existiert. Das kann nützlich sein, wenn Dokumentations-Dateien sowohl einzeln als auch kombiniert verarbeitet werden.

`doc`: Bessere Unterstützung für `upquote`

In den LaTeX News 37 [4, S. 89] haben wir geschrieben, dass das Paket `doc` um Unterstützung für das `upquote`-Paket erweitert wurde, aber damals galt das nur für `\verb` und die `verbatim`-Umgebungen. In einer typischen `.dtx`-Datei wird sich der größte Teil des Codes aber im Rumpf einer `macrocode`- oder `macrocode*`-Umgebung befinden und keine davon wurde durch das Hinzufügen von `upquote` verändert. Wir haben `doc` jetzt so angepasst, dass `upquote` die Zitierzeichen in diesem Umgebungen ebenfalls verändert. *(Github Issue 1230)*

`ifthen`: Schutz gegen aktive Zeichen in Vergleichen

Der Befehl `\ifthenelse` stellt nun sicher, dass ‹, = und › in numerischen Tests sicher sind, selbst wenn sie aktiv gemacht wurden (oft durch Sprachkürzel in `babel`). *(Github Issue 756)*

Neue Conditionals: `\IfClassAtLeastT` & Co

Ungefähr im Jahr 2020 haben wir eine Reihe von Conditionals mit CamelCase-Namen ergänzt, unter anderem

- `\IfClassAtLeastTF`,
- `\IfClassLoadedTF`,
- `\IfClassLoadedWithOptionsFF`,
- `\IfFormatAtLeastTF`,
- `\IfPackageAtLeastTF`,
- `\IfPackageLoadedTF` und
- `\IfPackageLoadedWithOptionsTF`.

Damit sollte es erleichtert werden, bedingt ausgeführten Code zu schreiben, der vom Release einer bestimmten Klasse, eines Pakets oder Formats abhängt. Aber wir haben nur die `TF`-Befehle und nicht auch die `T`- und `F`-Varianten bereitgestellt. Das haben wir nun geändert.

Im Jahr 2023 haben wir dann `\IfFileAtLeastTF` eingeführt, aber nicht gleichzeitig `\IfFileLoadedTF` mitgeliefert. Dieses Conditional und seine `T`- und `F`-Varianten wurden nun ebenfalls ergänzt. Denken Sie daran, dass man nur auf Dateien testen kann, die eine `\ProvidesFile`-Zeile enthalten.

Das Gleiche haben wir für `\IfLabelExistsTF` und `\IfPropertyExistsTF` umgesetzt, die ebenfalls 2023 hinzukamen.[4] *(Github Issues 1222 1262)*

Verbesserungen am Code

Pakete nur auf oberster Ebene laden

Klassen und Pakete dürfen nur über die Befehle `\documentclass` und `\usepackage` oder über die Befehle der Klassenschnittstelle geladen werden (wie `\LoadClass` oder `\RequirePackageWithOptions`), zudem müssen diese immer auf oberster Ebene geladen werden, nicht innerhalb einer Gruppe irgendeiner Art (zum Beispiel innerhalb eines Environments). Früher hat LaTeX das nicht überprüft, was oft zu Low-Level-Fehlern geführt hat, wenn Paketdeklarationen am Ende einer Gruppe automatisch rückgängig gemacht wurden. LaTeX prüft nun die Gruppenebene und stoppt mit einer Fehlermeldung, wenn die Klasse oder das Paket in einer Gruppe geladen wird. *(Github Issue 1185)*

Fehlende Glyphen im Auge behalten

Vor einiger Zeit haben wir den Standardwert von LaTeX für `\tracinglostchars` von `1` in `2` geändert, so dass fehlende Glyphen zumindest eine Warnung erzeugen. Aber wir haben vergessen, das Gleiche für `\tracingnone` umzusetzen. Nach Aufruf dieses Befehls hat LaTeX daher keine Warnungen mehr über fehlende Glyphen ausgegeben, was nun korrigiert wurde. *(Github Issue 549)*

`fontenc`-Fehlermeldung verbessert

Wenn das Paket `fontenc` gebeten wird, ein Font-Encoding zu laden, für das es keine passende `.def`-Datei findet, erzeugt es eine Fehlermeldung, die andeutet, dass

[4] Diese wurden fälschlicherweise zunächst unter den Namen `\IfLabelExistTF` und `\IfPropertyExistTF` eingeführt – das haben wir jetzt ebenfalls korrigiert. Dabei handelt es sich um einen Breaking Change, aber die Befehle wurden bisher unseres Wissens nur im Kernelcode verwendet.

der Encoding-Name falsch geschrieben sein könnte. Das ist natürlich nur einer der möglichen Gründe – ein anderer ist, dass bei der Installation ein notwendiges Support-Paket fehlt, zum Beispiel, dass keine Unterstützung für kyrillische Schriften installiert wurde. Der Text der Fehlermeldung wurde daher erweitert, um das Problem allgemeiner zu beschreiben. *(Github Issue 1102)*

Bei problematischen Zählernamen warnen

In der Vergangenheit war es möglich, beispielsweise `\newcounter{index}` zu deklarieren, wodurch als Nebeneffekt `\theindex` definiert wird, obwohl LaTeX eine `theindex`-Umgebung besitzt, die dann durch die Deklaration plattgemacht wird. Dies wurde nun geändert: Ist `\the`⟨*counter*⟩ schon definiert, wird es nicht verändert, sondern stattdessen eine Warnmeldung ausgegeben. *(Github Issue 823)*

Erweiterte Informationen in `\listfiles`

Der Befehl `\listfiles` stellt nützliche Informationen zur Verfügung, wenn man Probleme analysieren muss, die mit unterschiedlichen Paketversionen zu tun haben. Aber diese Informationen konnten bisher nur auf das zurückgreifen, was in der `\ProvidesPackage`-Zeile oder ähnlichem angegeben war. Das kann zum Beispiel in die Irre führen, wenn eine Datei lokal bearbeitet wurde. Wir haben `\listfiles` jetzt um ein optionales Argument erweitert, welches den MD5-Hash und die Größe jeder Datei in die `.log`-Datei übernehmen kann. Sie können nun also beispielsweise

```
\listfiles[hashes,sizes]
```

verwenden, um neben den Standard-Releaseinformationen sowohl die Datei-Hashes als auch die Dateigrößen in der `.log`-Datei zu erhalten. *(Github Issue 945)*

Optimiertes Erstellen einfacher Dokumentbefehle

Das Erstellen von Dokument-Befehlen über Deklarationen wie `\NewDocumentCommand` ermöglicht es sehr flexibel und einfach, Kommandos mit komplexer Argument-Syntax zu definieren. Geht es aber um einfache Kommandos mit nur obligatorischen Argumenten (in geschweiften Klammern) hatten so definierte Kommandos trotzdem einen deutlichen Overhead bei der Ausführung, der nicht zu vermeiden war. Wir haben nun den internen Codepfad so optimiert, dass »einfache« Dokumentbefehle fast ohne Overhead bei der Verwendung auskommen, so dass die Ergebnisse im Wesentlichen genauso effizient sind wie bei der Verwendung von `\newcommand` für Low-Level-TeX-Konstrukte.

Beachten Sie, dass `\NewDocumentCommand` Engine-robuste Befehle erstellt; das direkte Äquivalent zu `\newcommand` ist `\NewExpandableDocumentCommand`. *(Github Issue 1189)*

Umgang mit End-of-Lines in `+v`-Argumenten von `\NewDocumentCommand` & Co

Der von Deklarationen wie `\NewDocumentCommand` bereitgestellte Argumenttyp `+v` erlaubt das Übernehmen mehrerer Textzeilen in einem Verbatim-ähnlichen Argument. So gut wie immer werden diese Daten dann in einem Schriftsatzkontext eingesetzt. Bisher wurden die End-of-Line-Zeichen als `^^M`-Token der Kategorie Code 12 (»other«) übernommen. Damit lässt sich aber im Allgemeinen schlecht arbeiten. Wir haben dieses Verhalten nun überarbeitet, so dass End-of-Line-Zeichen beim Parsen von Argumenten des Typs `+v` in den Befehl `\obeyedline` umgewandelt werden. Durch diese Änderung sind eventuell Anpassungen an manchen Dokument-Sourcen erforderlich, aber die verbesserten Möglichkeiten für User und beim Programmieren rechtfertigen dies unserer Meinung nach.

Passende Sub-Kodierungen für `TS1`-Symbolfonts deklarieren

Im Jahr 2020 haben wir die Unterstützung für die `TS1`-Symbolkodierung direkt in den Kernel eingebaut und damit die Notwendigkeit beseitigt, das `textcomp`-Paket laden zu müssen [4, S. 53], um Befehle wie `\texteuro` verfügbar zu machen.

Es gibt aber ein großes Problem mit dieser `TS1`-Symbolkodierung: Nur sehr wenige Fonts stellen jeden Glyph bereit, der als Teil von `TS1` angesehen wird. Ein Wechsel der Schriftfamilie kann daher dazu führen, dass bestimmte Symbole nicht verfügbar sind. Das kann eine mittlere Katastrophe sein, wenn beispielsweise das Symbol `\texteuro` (€) oder `\textohm` (Ω) nicht mehr in Ihrem Dokument ausgegeben wird, nur weil Sie die Text-Schriftfamilie geändert haben.

Um dieses Problem abzumildern, haben wir im gleichen Jahr auch die Deklaration `\DeclareEncodingSubset` hinzugefügt. Diese sollte in allen Font-Definitionsdateien für die `TS1`-Kodierung genutzt werden, um anzugeben, welches Subset (wir haben zehn gebräuchliche definiert) ein bestimmter Font implementiert. Wird solch eine Deklaration genutzt, werden fehlende Symbole aus einem Fallback-Font genommen.

Das ist zwar nicht perfekt, aber das Beste, was man automatisiert tun kann, es sei denn, Sie prüfen akribisch, dass Ihr Dokument nur Glyphen nutzt, die der Font unterstützt, und wechseln andernfalls zu einem anderen Font oder vermeiden die fehlenden Symbole. Siehe dazu auch den Text in LaTeX News 33 [4, S. 65].

Um den Prozess zu beschleunigen, haben wir zudem den LaTeX-Kernel um Deklarationen für die meisten der Fonts ergänzt, die zu dem Zeitpunkt in TeX Live zu finden

waren – unter der Annahme, dass solche Deklarationen mit der Zeit durch Deklarationen in den `.fd`-Dateien überschrieben werden würden. Leider ist das noch nicht geschehen (oder zumindest nicht sehr oft) und daher sind viele der initialen Deklarationen veraltet: Diverse Fonts haben neue, zusätzliche Glyphen erhalten (so dass ihre Sub-Kodierung geändert werden sollte, was aber nicht geschah); andere (hauptsächlich wegen Lizenzproblemen) haben den Familiennamen geändert und damit unsere Deklarationen nutzlos gemacht, so dass die umbenannten Fonts (nun ohne Deklaration) in der Standard-Sub-Kodierung landeten, die nur wenige Glyphen bietet; wieder andere, wie CharisSIL (der das GitHub-Problem 1257 ausgelöst hat), waren einfach noch nicht verfügbar.

Wir haben daher erneut versucht, die (aktuell) korrekten Deklarationen bereitzustellen, aber es ist offensichtlich, dass dies kein praktikabler Prozess ist. Da wir die Fonts nicht pflegen, haben wir nicht die Informationen, dass sich etwas geändert hat, und es ist schlicht nicht möglich, regelmäßig die immer größer werdenden Font-Support-Bundles zu überprüfen. Es ist daher sehr wichtig, dass die Maintainer von Fontpaketen nicht nur `.fd`-Dateien bereitstellen, sondern auch eine solche Deklaration zu jeder `TS1...fd`-Font-Definitionsdatei hinzufügen, die sie verteilen.

Um den Prozess zu vereinfachen, stellen wir nun eine einfache LaTeX-Datei (`checkencodingsubset.tex`) bereit, mit der die korrekte (sichere) Sub-Kodierung ermittelt werden kann. Wenn man sie ausführt, fragt sie nach einer Fontfamilie und gibt dann ihre Ergebnisse aus. Für `AlgolRevived-TLF` erhalten Sie beispielsweise:

```
----------------------------------------
Testing font family AlgolRevived-TLF
(currently TS1-sub-encoding 9)
----------------------------------------
Some glyphs are missing from sub-encoding 8:
   ==> \textcelsius (137) is missing
   ==> \texttwosuperior (178) is missing
   ==> \textthreesuperior (179) is missing
   ==> \textonesuperior (185) is missing
Some glyphs are missing from sub-encoding 7:
   ==> \texteuro (191) is missing
All glyphs between sub-encoding 6 and 7 exist
All glyphs between sub-encoding 5 and 6 exist
All glyphs between sub-encoding 4 and 5 exist
Some glyphs are missing from sub-encoding 3:
   ==> \textwon (142) is missing
All glyphs between sub-encoding 2 and 3 exist
Some glyphs are missing from sub-encoding 1:
   ==> \textmho (77) is missing
   ==> \textpertenthousand (152) is missing
```

```
All glyphs between sub-encoding 0 and 1 exist
All glyphs in core exist
----------------------------------------
TS1 encoding subset for AlgolRevived-TLF (ok)
Use sub-encoding 9
----------------------------------------
```

Diese Ausgabe ist für menschliche Augen gedacht – Sie sehen, welche Glyphen fehlen und warum eine bestimmte Sub-Kodierung vorgeschlagen wird. Aber allzu schwer sollte der Einsatz in einem Skript auch nicht sein, wenn man die vorgeschlagenen Sub-Kodierungen extrahiert, indem man die Zeile greppt, die mit `Use sub-encoding` beginnt.

Natürlich wird dieser Check nur funktionieren, wenn die fehlenden Glyphen wirklich fehlen. Bei manchen Fonts ist an solchen Stellen »Tofu«[5] und in diesem Fall sieht es für TeX so aus, als ob die Glyphe vorhanden ist. Für die alten Palatino-Fonts (`ppl`-Familie) würde beispielsweise folgendes berichtet werden;

```
----------------------------------------
TS1 encoding subset for ppl (bad)
Use sub-encoding 0 (not 5)
----------------------------------------
```

Angeblich sind also alle Glyphen vorhanden, aber in Wirklichkeit fehlen mehr als 20 und die Sub-Kodierung 5 – wie im Kernel deklariert – ist tatsächlich korrekt.

(Github Issue 1257)

Verhalten beim Laden von `textcomp` ohne Optionen

Beim Einbinden des Pakets `textcomp` in den LaTeX-Kernel im Release vom Februar 2020 [4, S. 51] wurde der Standardtyp seiner Paketnachrichten von Package-Info (`Package textcomp Info`) in LaTeX-Kernel-Info (`LaTeX Info`) geändert. Wurde `textcomp` aber ohne Optionen geladen, wurde der Nachrichtentyp wieder auf Package-Info zurückgesetzt. Dieses Zurücksetzen wurde nun unterbunden.

Beachten Sie, dass Laden von `textcomp` mit einer der Optionen `error`, `warn` oder `info` weiterhin den Nachrichtentyp von Nachrichten aus diesem Paket zu einer Fehler-, Warn- oder Infonachricht ändert. *(Github Issue 1333)*

Verbesserungen zurückrollen

Beim Anfordern eines Rollbacks des LaTeX-Kernels und/oder von Paketen haben eine Reihe von Paketen den Fehler »Suspicious rollback date« gemeldet, weil ihr

[5] Kleine Quadrate, die für ein fehlendes Symbol stehen.

Rollback-Bereich nur Daten über aktuelle Releases enthielt, auch wenn das Paket, wie `array`, seit der ersten Veröffentlichung von LaTeX 2_ε im Jahr 1994 verfügbar war. Wir unterdrücken diesen Fehler nun und laden die erste Version, die noch Teil der Distribution ist (und drücken die Daumen). Diese Änderung wurde für die Pakete `amsmath`, `array`, `doc`, `graphics`, `longtable`, `multicol`, `showkeys`, `textcomp` und `varioref` umgesetzt. *(Github Issue 1333)*

Verbesserungen der Dokumentation

Weitere Aktualisierungen der Guides

Wir haben in den LaTeX News 37 [4, S. 91] über die aktualisierten Versionen von `usrguide` und `clsguide` berichtet. Nun wurde ebenfalls `fntguide` überarbeitet, um die Änderungen und zusätzlichen Makros im Kernel der letzten Jahre zu berücksichtigen. Beachten Sie, dass sich der Dateiname nicht geändert hat und es keine `fntguide-historic.pdf`-Datei gibt.

Fehlerkorrekturen

Inkonsistente Expansion der Paketoptionen-Liste korrigiert

LaTeX expandiert die Optionsliste von Paketen und Klassen einmalig, so dass Konstrukte wie

```
\def\myoptions{opt1,opt2}
\usepackage[\myoptions]{foo}
```

unterstützt werden. Deklariert ein Paket seine Optionen aber mit Hilfe des neuen Schlüssel/Wert-Ansatzes (beschrieben in LaTeX News 35 [4, S. 77]) und wird es ein zweites Mal geladen, wird Optionsliste dann nicht expandiert und es kann zu einer Fehlermeldung kommen. Das wurde nun korrigiert. *(Github Issue 1298)*

Logik für die erste Marke (Seitenregion) korrigieren

Im neuen, im Juni 2022 eingeführten Markierungsmechanismus (beschrieben in LaTeX News 35 [4, S. 76]) war das Ergebnis von `\FirstMark` bei einer zweispaltigen Seite nicht korrekt, wenn die erste Spalte keine Markierungen enthielt. In diesem Fall sollte die erste Markierung der zweiten Spalte zurückgegeben werden, was aber nicht geschah. Das wurde nun korrigiert.

Dokumente, die `\leftmark` verwenden, sind nicht betroffen, weil dieser Befehl bisher immer noch den alten Mechanismus nutzt. *(Github Issue 1359)*

Struts am Ende von Fußnoten oder p-Spalten

Um in Fußnoten und Tabellen-p-Feldern ein konsistentes Spacing zu erreichen, fügt LaTeX am Anfang und Ende des Inhalts einen Strut hinzu. Dabei wird allerdings davon ausgegangen, dass der Inhalt der Fußnote oder des Tabellenfeldes im horizontalen Modus endet und daher wurden diese Struts bisher immer ergänzt. Endete der Inhalt mit vertikalem Material, sorgten diese Struts für einen neuen Absatz mit einer einzelnen Zeile, die nur den Strut enthielt. Das wurde nun endlich korrigiert und die Positionierungslogik für den Strut ändert sich, wenn der vertikale Modus erkannt wird.

(Zuerst in einem Bug Report für `footmisc` in Kombination mit `bigfoot` gesehen)

Beheben eines »missing \item« Rollback-Fehlers

Wurde LaTeX auf ein Datum zwischen 2023/06/01 (inklusive) und 2024/06/01 (exklusive) zurückgerollt, warfen alle listenbasierten Umgebungen den Fehler:

```
! LaTeX Error: Something's wrong--perhaps a missing \item.
```

Das wurde nun als Hotfix in Patch Level 2 korrigiert, in dem der Rollback-Code von Version 2023/06/01 des neuen Absatzmechanismus erweitert wurde. *(Github Issue 1386)*

Änderungen an Paketen in der Kategorie amsmath

amsmath: Korrekte Positionierung der Gleichungsnummer

Ist nicht ausreichend Platz, um die Gleichungsnummer in der gleichen Zeile wie die Gleichung unterzubringen, berechnet amsmath einen passenden Offset und platziert die Nummer oberhalb (oder unterhalb) der Gleichung. Bei der gather-Umgebung wurde dieser Offset am Ende nicht zurückgesetzt, was dazu führte, dass er auch in allen folgenden Umgebungen angewendet wurde, so dass es in bestimmten Situationen zu einem falschen Spacing kam. Das wurde nun korrigiert. *(Github Issue 1289)*

Änderungen an Paketen in der Kategorie tools

array, longtable, tabularx: Tagging-Unterstützung

Diese drei Pakete wurden so erweitert, dass sie nun auf Anforderung getaggte Tabellen erzeugen können. Das geschieht durch das Hinzufügen einer Reihe von Sockets (siehe LaTeX News 38 [4, S. 93]), die standardmäßig nichts tun – wird getaggtes PDF angefordert, erhalten sie die passenden Plugs.

Im vorigen LaTeX-Release wurde das in `latex-lab` umgesetzt, indem die Pakete gepatcht wurden, wenn Tagging gewünscht war.

`array`: Kein `\unskip` in Mathematik-Feldern

Mathematik-Felder in der Standard-`array`-Umgebung des Kernels entfernen keinen Leerraum am rechten Ende des Feldes, zum Beispiel wird expliziter Leerraum von `\hspace` oder `\␣` etc. berücksichtigt (normaler Leerraum wird im Mathematik-Modus automatisch ignoriert). Im Paket `array` wurde durch den bedingungslosen Aufruf von `\unskip` jeglicher Leerraum entfernt – unabhängig von der Art des Feldes. Dieser Unterschied im Verhalten wurde nun korrigiert, indem die Verarbeitung von Mathematik-Feldern in `array` angepasst wurde. *(Github Issue 1323)*

`verbatim`: `\verb` hat sichtbare Leerzeichen gezeigt

Eine aktuelle Änderung im Kernel wurde nicht im `verbatim`-Paket berücksichtigt, weshalb `\verb` sichtbare Leerzeichen (␣) anzeigte, nachdem das Paket geladen wurde. Dies wurde bereits in einem Hotfix für das Release im November 2023 korrigiert. *(Github Issue 1160)*

`verbatim`: Unterstützung von Tabulatoren in `\verbatiminput*`

Das Paket `verbatim` wurde so angepasst, dass es die Aktualisierung des Kernels aus dem Jahr 2023 nachbaut, welches mit `\verb*` Tabulatoren als Leerzeichen anzeigt. Nun zeigt `\verbatiminput*` ebenfalls Tabulatoren als Leerzeichen an. *(Github Issue 1245)*

`multicol`: `\columnbreak` kommt dem Mechanismus für lebendige Seitentitel in die Quere

Das Paket `multicol` muss als Teil seiner Ausgaberoutine Seitentitel im Blick behalten (aus `\markright` oder `\markboth`) und kann sich nicht darauf verlassen, dass LaTeX das automatisch macht. Dazu teilt es die Seite künstlich mit `\vsplit` auf, um die Seitentiteldaten zu extrahieren. Mit der Einführung von `\columnbreak` schlug dieser Code manchmal fehl, weil er keine Seitentitel sah, die solch einem erzwungenen Spaltenumbruch folgten.

Das wurde nun korrigiert, aber die Arbeit ist noch nicht erledigt, weil `multicol` Seitentitel aktuell noch nicht mit dem neuen Auszeichnungsmechanismus verarbeitet – siehe die Diskussion zu Beginn dieses Newsletters. *(Github Issue 1130)*

showkeys: \newline in amsthm funktioniert

Bisher hat showkeys einen zusätzlichen Box Layer hinzugefügt, der die \newline von amsthm-Theorem-Stilen deaktivierte. Diese zusätzliche Box wird nun vermieden. *(Github Issue 1123)*

xr: Unterstützung von Links und Eigenschaften

Das Paket xr implementiert ein System für eXterne Referenzen. Das Paket xr–hyper (im hyperref-Bundle) erweitert dies so, dass auch Links auf externe Dokumente unterstützt werden. Mit der Erweiterung des Befehls \label aus dem letzten Jahr, die die Label-Syntax von LaTeX und hyperref vereinheitlicht hat, wurde es nun möglich, die zwei Pakete zusammenzuführen und xr–hyper damit obsolet zu machen. Durch diese Änderung ist es nun zudem möglich, sich auf Eigenschaften zu beziehen, die in externen Dokumenten mit \RecordProperties abgelegt sind. *(Github Issue 1180)*

Änderungen an Dateien der Kategorie cyrillic

Korrekte Definition von \k

Vor vielen Jahren wurden die kodierungsspezifischen Definitionen für eine Reihe von Akzent-Befehlen geändert, um zu verhindern, dass bestimmte Parameter unabsichtlich global geändert wurden. Aus irgendeinem Grund wurde das bei der Definition von \k in den kyrillischen Kodierungen T2A, T2B und T2C unterlassen. Dies wurde nun korrigiert. *(Github Issue 1148)*

Literatur

[1] Leslie Lamport: LaTeX: A Document Preparation System: User's Guide and Reference Manual, 2nd edition, Reprinted with corrections in 1996, Addison-Wesley, Reading, MA, USA, 1994, ISBN: 0-201-52983-1.

[2] Frank Mittelbach, Ulrike Fischer: »Enhancing LaTeX to automatically produce tagged and accessible PDF«, *TUGboat*, *45.1* (2024), https://latex-project.org/publications/indexbyyear/2024/.

[3] »LaTeX News 39«, *TUGboat*, *45.2* (2024), 240–245, ISSN: 0896-3207, https://tug.org/TUGboat/tb45-2/tb140ltnews39.pdf.

[4] LaTeX Project Team: LaTeX 2ε News 1–39, https://latex-project.org/news/latex2e-news/ltnews.pdf.

Neue Pakete auf CTAN

Jürgen Fenn

Der Beitrag stellt neue Pakete auf CTAN seit der letzten Ausgabe bis zum Redaktionsschluss in umgekehrter chronologischer Reihenfolge vor. Bloße Updates können auf der moderierten *CTAN-ann*-Mailingliste oder als RSS-Feed auf `https://ctan.org/` verfolgt werden.

`beamertheme-mirage` von *LianTze Lim* wurde von der Gestaltung des Albums für den Song *Mirage* von *Zhou Shen* inspiriert. Das Theme ist in einer dunklen und in einer hellen Version verfügbar.
`CTAN:macros/latex/contrib/beamer-contrib/themes/beamertheme-mirage`

`sshrc-insight` von *Tristan Miller* ist eine Dokumentenklasse für Forschungsanträge für das *Insight Grants program* des kanadischen *Social Sciences and Humanities Research Council* (SSHRC).
`CTAN:macros/latex/contrib/sshrc-insight`

`temporal-logic` von *Dominik Schmid* stellt Befehle bereit, mit denen man Operatoren aus dem Bereich der Temporalen Logik setzen kann, mit denen man also zeitliche Beziehungen zwischen mehreren Ausdrücken oder allgemein zeitliche Umstände einer Aussage zum Ausdruck bringt.
`CTAN:macros/latex/contrib/temporal-logic`

`sunpath` von *Hồng-Phúc Bùi* kann Grafiken zum Sonnenstand mit Hilfe von pgf/Ti*k*Z zeichnen.
`CTAN:graphics/pgf/contrib/sunpath`

`numbersets` von *Takumi Noguchi* stellt ein Interface bereit, mit dem man die Formelzeichen für Zahlenmengen generisch auszeichnen kann, um sie in einem Dokument konsistent zu bezeichnen und zu setzen. Es arbeitet standardmäßig ohne das Paket `unicode-math`, kann aber auch damit verwendet werden.
`CTAN:macros/latex/contrib/numbersets`

`tikz-decofonts` von *Cédric Pierquet* enthält drei Schmuckschriften, die mit Ti*k*Z erstellt wurden: Es gibt eine Schrift, die aussieht wie ein Paint Brush, eine wie ein Tuschepinsel und einen Pixelfont. Sie eignen sich für kurze Texte und für Überschriften.
`CTAN:graphics/pgf/contrib/tikz-decofonts`

`mp-geom2d` von *Maxime Chupin* dient zum Zeichnen von geometrischen Figuren in der Ebene mit METAPOST. Es befindet sich noch in Entwicklung.
`CTAN:graphics/metapost/contrib/macros/mp-geom2d`

faq-fr-gutenberg von der *Association GUTenberg* enthält den Markdown-Quelltext und die PDF-Version der neuen französischsprachigen TEX-FAQ. Eine für das Web aufbereitete Fassung der *FAQ LATEX francophone* zu allem, was man schon immer über TEX, LATEX und alle weiteren Tools erfahren wollte, findet man unter `https://faq.gutenberg-asso.fr/`.
`CTAN:help/faq-fr-gutenberg`

hduthesis von *Mingyu Xia* ist eine Dokumentenklasse für Abschlussarbeiten an der chinesischen Hangzhou Dianzi Universität.
`CTAN:macros/xetex/latex/hduthesis`

udiss von der *Indian TEX Users Group* erleichtert das Abfassen von Doktorarbeiten, unabhängig von einer bestimmten Hochschule. Es ist zugleich das erste Projekt, das die nach 32 Jahren wiedergegründete Anwendervereinigung *TUGIndia* hervorgebracht hat.
`CTAN:macros/unicodetex/udiss`

fontscripts von *Clea F. Rees* automatisiert das Erstellen von Unterstützungspaketen zu Fonts für 8-bittige TEX-Engines mit `l3build`. Ursprünglich wurde das Paket für die Verwendung mit `fontinst` entwickelt, man kann es aber auch unabhängig davon auf allen Plattformen einsetzen, weil nur auf Tools zurückgegriffen wird, die in TEX Live enthalten sind.
`CTAN:fonts/utilities/fontscripts`

commalists-tools von *Cédric Pierquet* stellt Befehle bereit, mit denen man kommaseparierte Listen mit numerischen Werten verarbeiten kann (sortieren, durchzählen und Bestimmen der Anzahl der Zahlen in der Liste, addieren und multiplizieren, Durchschnittswerte berechnen und einiges mehr). Das Paket funktioniert mit allen TEX-Engines.
`CTAN:macros/latex/contrib/commalists-tools`

randintlist von *Cédric Pierquet* erzeugt eine bestimmte Anzahl von Integer-Zufallszahlen aus einem bestimmten Intervall, die man als konfigurierbare Liste ausgeben lassen kann. Es ähnelt dem Paket `luarandom`, von dem es sich aber dadurch unterscheidet, dass es auch ohne LuaLATEX funktioniert.
`CTAN:macros/latex/contrib/randintlist`

beamertheme-gotham von *Romain Noël* ist ein minimalistisches Theme, das auf dem `beamertheme-gotham` von *Matthias Vogelgesang* und auf LATEX3 beruht.
`CTAN:macros/latex/contrib/beamer-contrib/themes/beamertheme-gotham`

ezedits von *Joseph A. Smiga* dient dazu, die Beiträge von mehreren Autoren eines LATEX-Dokuments als solches auszuzeichnen und somit nachvollziehbar zu machen.
`CTAN:macros/latex/contrib/ezedits`

zugferd von *Marei Peischl* stellt Muster für elektronische Rechnungen nach der ZUGFeRD-Spezifikation bereit. Das Paket befindet sich noch in Entwicklung. Es enthält auch ein Muster für eine elektronische Rechnung nach dem Standard XRechnung.
`CTAN:macros/latex/contrib/zugferd`

cascadiamono-otf von *Cédric Pierquet* enthält die XeLaTeX/LuaLaTeX-Unterstützung für die OpenType-Schrift CascadiaMono, eine Variante von CascadioCode ohne Ligaturen.
`CTAN:fonts/cascadiamono-otf`

keytheorems von *Matthew Bertucci* ist ein l3keys-Interface zu `amsthm`. Es setzt die meisten Features aus dem Paket `thmtools` um und behebt zudem einige seiner bekannten Probleme.
`CTAN:macros/latex/contrib/keytheorems`

euromoney von *Cédric Pierquet* stellt Grafiken von Euro-Münzen und -Banknoten bereit, die in drei Ausführungen (farbig, vereinfacht farbig und vereinfacht schwarzweiß) vorliegen. Man kann sie einzeln oder in Stapeln zu mehreren in ein Dokument einfügen.
`CTAN:macros/latex/contrib/euromoney`

aiplans von *Yikai Ge* zeichnet Pläne als *Partial Order Causal Link* (POCL), die im Bereich der Künstlichen Intelligenz verwendet werden.
`CTAN:graphics/pgf/contrib/aiplans`

quran-es von *Seiied-Mohammad-Javad Razavian* enthält spanische Übersetzungen für das Paket `quran`.
`CTAN:macros/xetex/latex/quran-es`

scrhack von *Markus Kohm* ist eine weitere Ausgliederung aus dem Paket `koma-script`, dessen Quelltext schon seit längerem umgebaut und dabei bereinigt wird. Das Paket wird für zukünftige Versionen von `koma-script` nicht mehr benötigt, stellt aber bis auf weiteres die Kompatibilität zu älteren Versionen sicher.
`CTAN:macros/latex/contrib/scrhack`

framedsyntax von *Claudio Beccari* dienen dazu, (LaTeX-Quelltexte in schöne Boxen zu setzen, wie man es für eine Paketdokumentation braucht. Der Autor hatte das Paket entwickelt, um damit Anleitungen für die Mitglieder der italienischen TeX-Anwendervereinigung GuIT zu verfassen.
`CTAN:macros/latex/contrib/framedsyntax`

csthm von *Agni Datta* stellt konfigurierbare Theorem-Umgebungen für Texte aus der Informatik zur Verfügung.
`CTAN:macros/latex/contrib/csthm`

beamertheme-edmaths von *Josh Fogg* ist ein Beamer-Theme für Präsentationen der Universität Edinburgh.
CTAN:macros/latex/contrib/beamer-contrib/themes/beamertheme-edmaths

suanpan-l3 von *Nan Geng* ermöglicht das Zeichnen von Abakussen mit l3draw.
CTAN:macros/latex/contrib/suanpan-l3

xreview von *Lorenzo Peri* ist ein Paket, mit dem man Änderungen beim Überarbeiten eines Dokuments im Quelltext mit LaTeX-Befehlen auszeichnen kann. Ergänzungen, Streichungen und Anmerkungen kann man in den Quelltext als solche einfügen und beim Setzen des Dokuments ein- oder ausschalten.
CTAN:macros/latex/contrib/xreview

enverb von *Jonathan P. Spratte* ist ein weiteres Paket, mit dem man einen beliebigen Text buchstabengetreu setzen kann.
CTAN:macros/latex/contrib/enverb

interlinear von *Akpoué Kouamé Josué* dient zum Setzen von Interlinearglossierungen. Es beruht auf dem Paket gb4e von *Alexis Dimitriadis*, *Hans-Peter Kolb* und *Craig Thiersch* und kann zusammen mit linguex von *Wolfgang Sternefeld* verwendet werden.
CTAN:macros/latex/contrib/interlinear

domaincoloring von *Herbert Voß* dient zum Zeichnen komplexer Funktionsgraphen als *domain colouring* bzw. *colour wheel graph* mit Hilfe von LuaLaTeX.
CTAN:macros/luatex/latex/domaincoloring

edmaths von *Josh Fogg* ist die Dokumentenklasse für Abschlussarbeiten an der Universität Edinburgh.
CTAN:macros/latex/contrib/edmaths

luamml von *Marcel Krüger* ist eine experimentelle Lösung, mit der man mathematische Ausdrücke aus LuaLaTeX-Dokumenten nach MathML wandeln kann.
CTAN:macros/luatex/latex/luamml

hebdomon von *Daniel T. McGuiness* ist eine Dokumentenklasse, die für das Abfassen von Berichten in einem Hochschulumfeld entwickelt wurde.
CTAN:macros/latex/contrib/hebdomon

fillwith von *Clea F. Rees* füllt vertikalen Leerraum in Arbeitsblättern oder in Formularen mit einer einfachen Lineatur als Schreibhilfe, auf Wunsch auch mit Punkten. Das Paket geht auf eine Diskussion in TeX StackExchange zurück, die teilweise in expl3 neu geschrieben und erweitert wurde.
CTAN:macros/latex/contrib/fillwith

mfb-oldstyle von *Daniel Benjamin Miller* enthält die Serifenschrift *MFB Oldstyle* in den Formaten OpenType und Type 1, einschließlich der zugehörigen LaTeX-Unterstützung. Vorbild war die *Century Oldstyle* von *Morris Fuller Benton*, die im

Jahr 1909 von *American Type Founders* veröffentlicht wurde.
`CTAN:fonts/mfb-oldstyle`

polyomino von *Matthias Floré* zeichnet Polyominos in verschiedenen Varianten, auch selbst konfigurierbar, mit Hilfe von Ti*k*Z. Das sind Flächen, die aus Quadraten zusammengesetzt sind, wie man es zum Beispiel aus dem Computerspiel Tetris kennt.
`CTAN:graphics/pgf/contrib/polyomino/`

bib2qr von *Christian Schreinemachers* erstellt QR-Codes aus BibTeX-Einträgen, die einen Weblink zum DOI oder hilfsweise zur URL enthalten. Dadurch soll der Zugriff auf den Volltext erleichtert werden. Es setzt auf den Paketen `biblatex` und `qrcode` auf.
`CTAN:macros/latex/contrib/bib2qr`

Spielplan

Termine

3. 4. – 5. 4. 2025 **DANTE 2025**
h_da Hochschule Darmstadt
Gebäude C10
Schöfferstraße 3
64295 Darmstadt
https://www.dante.de/veranstaltungen/dante2025/

18. 7. – 20. 7. 2025 **TUG 2025**
Trivandrum, Indien
veranstaltet von TEXFolio
https://www.tug.org/tug2025/

23. 8. – 29. 8. 2025 **19th ConTEXt Meeting**
Ferienanlage Krefta bei Gdańsk, Polen
https://meeting.contextgarden.net/2025/

Stammtische

In verschiedenen Städten im Einzugsbereich von DANTE e.V. finden regelmäßig Treffen von TEXies statt, die für alle offen sind. Im Web gibt es aktuelle Informationen unter https://www.dante.de/dante-e-v/stammtische/.

Aachen
Torsten Bronger
bronger@physik.rwth-aachen.de
Mailingliste: https://lists.rwth-aachen.de/postorius/lists/tex-stammtisch.lists.rwth-aachen.de
»Anvers«, Kockerellstr. 20, 52062 Aachen
Erster Donnerstag im Monat, 20:00 Uhr

Bad Doberan
Carsten Vogel
texnicer@web.de
zur Zeit inaktiv, Interessenten bitte per Mail melden

Berlin
Michael-E. Voges, Tel.: 0 33 62/ 50 18 35,
mevoges@t-online.de
»La Esperanza Restaurant Tapas Bar«, Chausseestr. 131 B, 10115 Berlin
Zweiter Dienstag im Monat, 19:00 Uhr

Darmstadt
Karlheinz Geyer
geyerk@posteo.de
»Poseidon«, Rheinstr. 41, 64293 Darmstadt
Erster Freitag im Monat, 19:00 Uhr

Erlangen
Peter Seitz
p.seitz@KplusS-Ing.de
https://www.ks-ingenieurconsult.de/TeX/Stammtisch.html
Gaststätte »Deutsches Haus«, Luitpoldstr. 25, 91052 Erlangen
Dritter Dienstag im Monat, 19:00 Uhr

Frankfurt a. Main
Harald Vajkonny
vajkonny@t-online.de
zur Zeit inaktiv, Interessenten bitte per Mail melden

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
Günther Zander
guenther.zander@lug-balista.de
zur Zeit inaktiv. Bei Fragen steht Günther gern per Mail zur Verfügung.

Hannover
Reiko Kaps
kaps@luis.uni-hannover.de, *RRZN/LUIS, 3D-Raum, Schloßwender Str. 6 (Gebäude 1210), 30159 Hannover*
Zweiter Donnerstag im Monat, 18:30 Uhr, Terminabsprache über Mailingliste

Heidelberg
Martin Wilhelm Leidig, Tel.: 01 70 41 83 32 9,
moss@moss.in-berlin.de
Anmeldeseite zur Mailingliste: https://tinyurl.com/stammtisch-HD
wechselnder Ort, i. d. R. letzter Freitag im Monat

Köln
Uwe Ziegenhagen
uwe@dante.de, *zur Zeit inaktiv, Interessenten bitte per Mail melden*

Leipzig
Erhard Pross
Erhard.Pross@gmx.de
le-tex publishing services GmbH, Weißenfelser Str. 84, 04229 Leipzig

München
Leah Neukirchen
leah@vuxu.org
https://stammtisch-muenchen.dante.de/
Erste Woche in geradzahligen Monaten, wechselnde Wochentage und Orte

Stuttgart
Bernd Raichle
bernd.raichle@gmx.de
»Trollinger« beim Feuersee, Rotebühlstr. 50, 70178 Stuttgart
Zweiter Dienstag im Monat, 19:30 Uhr

Adressen

DANTE, Deutschsprachige Anwendervereinigung TEX e.V.
Bergheimer Straße 147
69115 Heidelberg

Tel.: (0 62 21) 2 97 66
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:	Uwe Ziegenhagen	vice-president@dante.de
Schatzmeisterin:	Doris Behrendt	treasurer@dante.de
Schriftführer:	Volker RW Schaa	secretary@dante.de
Beisitzer:	Klaus Höppner	
	Harald König	
	Stephan Lukasczyk	
	Oliver Rath	

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:	https://www.dante.de/	(Erik Braun)
CTAN:	https://mirror.ctan.org/	(Gerd Neugebauer)
DANTE-EV:	https://lists.dante.de/mailman/listinfo/dante-ev	

FAQ

DTK:	https://wiki.dante.de/dtk
TEX:	https://wiki.dante.de/dantefaq

TEXnische Fragen

beraterkreis@dante.de
ak-schule@dante.de

Autoren/Organisatoren

Luzia Dietsche [3,41]
71394 Kernen
dtkred@dante.de

Jürgen Fenn [66]
Neu-Isenburg
juergen.fenn@gmx.de

Frank Mittelbach [52]
LaTeX Project Team
Mainz
frank.mittelbach@latex-project.org

Henning Hraban Ramm [42]
hraban@fiee.net

Volker RW Schaa [9]
Bismarckstr. 21a
64293 Darmstadt
volker@dante.de

Martin Sievers [4,6,8]
siehe Seite 74

Torsten-Karl Strempel [6,8]
torsten-karl.strempel@h-da.de

Keno Wehr [18]
wehr@abgol.de

Dr. Alexander Willand [34]
Karlsruhe
alexander.willand@t-online.de

Uwe Ziegenhagen [48]
Köln

Die TEXnische Komödie

36. Jahrgang Heft 4/2024 November 2024

Impressum

Editorial

Hinter der Bühne

4 Grußwort
6 Einladung zur Frühjahrstagung 2025 und 67. MV in Darmstadt
8 Beiträge gesucht (»Call for Presentations«)
9 Bericht von der TUG 2024 in Prag — Tag 1

Bretter, die die Welt bedeuten

18 LaTeX und Schulphysik 8: Atom- und Kernphysik
34 Variable Fonts
41 Aufgeschnappt
42 ConTEXt kurz notiert
48 ASN-Labels mit LaTeX

Von fremden Bühnen

52 LaTeX News – Issue 39, June 2024
66 Neue Pakete auf CTAN

Spielplan

71 Termine
72 Stammtische

Adressen

75 Autoren/Organisatoren