Die TEXnische Komödie

dante
Deutschsprachige
Anwendervereinigung TEX e.V.

34. Jahrgang Heft 1/2022 Februar 2022

1/2022

Impressum

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 1.13.2 (TeX Live 2021)` erstellt. Als Standardschriften kamen Libertinus Serif, Libertinus Sans Serif, Anonymous Pro und Libertinus Math zum Einsatz.

Erscheinungsweise: vierteljährlich
Erscheinungsort: Heidelberg
Auflage: 2400
Herausgeber: DANTE, Deutschsprachige Anwendervereinigung TEX e.V.
Postfach 11 03 61
69072 Heidelberg
E-Mail: `info@dante.de` (DANTE e.V.)
`dtkred@dante.de` (Redaktion)
Druck: Schleunungdruck GmbH
Eltertstraße 27, 97828 Marktheidenfeld
Redaktion: Luzia Dietsche (verantwortliche Redakteurin)
Mitarbeit: Adelheid Bonnetsmüller, Rudolf Herrmann, Eberhard Lisse, Ralf Mispelhorn, Rolf Niepraschk, Stefan Pinnow, Bernd Raichle, Christine Römer, Herbert Voß

Redaktionsschluss für Heft 2/2022: 15. April 2022 ISSN 1434-5897

Editorial

Liebe Leserinnen und Leser,

in dieser Ausgabe von *Die TEXnische Komödie* gibt es ein sehr wichtiges Datum, über das ich mich ungemein freue – wenn nichts dazwischen kommt, wird im Juni die diesjährige Frühjahrstagung von DANTE e.V. stattfinden. Ich fand zwar auch schon die online durchgeführte Tagung im letzten Jahr ausgesprochen gelungen, aber ein Treffen mit direktem Gegenüber ist doch um ein Vielfaches erfreulicher. Endlich wieder Freunde treffen, interessante Menschen kennenlernen, über spannende Themen direkt diskutieren, auf unbequemen Hörsaalplätzen sitzend einem Vortrag lauschen und sich abends bis zur Sperrstunde austauschen können. Ich drücke uns sämtliche Daumen, dass es klappt!

Von all den Artikeln in diesem Heft ist für mich der Interessanteste sicher der von Adelheid Bonnetsmüller, behandelt er doch ein Thema, dem ich von zwei Seiten etwas abgewinnen kann – als Strickerin *und* als LaTeX-Anwenderin. Es fasziniert mich immer wieder, was man mit TeX alles umsetzen kann. Ich habe zwar nicht vor, eine eigene Anleitung zu schreiben, aber ich habe zum Stricken einen sehr viel direkteren Bezug, als beispielsweise zu VSCodium. Ich muss zu meiner Schande gestehen, dass ich bis dato noch nicht einmal davon gehört hatte. Aber das ist einer der Vorteile bei meiner Arbeit mit der Mitgliederzeitung: Mit jedem Mal lerne ich etwas Neues dazu. Ich hoffe sehr, dass es auch Ihnen so geht und Sie neue Erkenntnisse zu Tabellen, Seitenzählern, dem Erstellen von Kalendern und/oder dem Definieren von Präsentationen in XML erhalten.

So wünsche ich Ihnen und Euch wieder viel Spaß beim Lesen und
verbleibe mit TEXnischen Grüßen
Luzia Dietsche

Hinter der Bühne

Vereinsinternes

Grußwort

Liebe Mitglieder,

ich hoffe, Sie sind alle gut und vor allem gesund ins neue Jahr gestartet. Die ersten Wochen und das ein oder andere Ereignis liegen schon wieder hinter uns.

Wir haben uns im Vorstand zu Beginn des Jahres natürlich gefragt, ob das neue Jahr im Bezug auf unsere Vereinsaktivitäten so werden wird wie die vergangenen beiden oder ob es Chancen bietet, etwas Neues (oder Altes?) zu probieren.

Wir wollen etwas wagen und haben deshalb den bewährten Jahreskalender ein wenig umgeworfen. Es soll in diesem Jahr endlich wieder die Möglichkeit geben, an einer Tagung von DANTE e.V. in Präsenz teilzunehmen. Gleichzeitig nehmen wir die allgemeine Lage ernst und werden auch die digitale Teilnahme im Blick behalten.

Die Entscheidung für ein solches Konzept greift aus Sicht des Vorstands aber nur, wenn man einen entsprechenden Termin für die Frühjahrstagung wählt. Daher haben wir entschieden, so weit wie möglich Richtung Sommer zu gehen. Herausgekommen ist fast genau die Mitte des Jahres, der 23.–25. Juni. Die Einladung zu der Tagung finden Sie in dieser Ausgabe der Mitgliederzeitung auf Seite 6.

Bei der Mitgliederversammlung will der Vorstand zu einem den pandemiebedingten »Rückstau« bei den Berichten der Rechnungsprüfer:innen auflösen. Zum anderen bitten wir im Hinblick auf zukünftige Tagungen um eine Satzungsänderung, die dem Vorstand die grundsätzliche Möglichkeit einräumt, neben den klassischen Präsenztagungen auch Hybrid- oder in Ausnahmefällen, wie wir sie zuletzt erlebt haben, rein digitale Tagungen durchführen zu können.

Ich glaube, es könnte für diesen »Neuanfang« keinen besseren Partner vor Ort geben als Mathias Magdowski. Er hat uns bereits im letzten Jahr virtuell in Magdeburg begrüßt und wird es dieses Jahr dann hoffentlich vor Ort bzw. hybrid tun können. Wer seine zahlreichen Beiträge bei Twitter, Twitch, Wordpress und Co. verfolgt,

bekommt einen guten Eindruck davon, wie gut Mathias die Möglichkeiten des Digitalen nutzt und mittlerweile eben auch in hybride Veranstaltungen integriert.

Ich möchte mich schon heute ganz herzlich bei ihm für seine Bereitschaft bedanken, uns mit seinem Know-how zu unterstützen und hoffe sehr, dass wir es ihm auch mit einer großen Teilnehmerzahl danken können.

Traditionell entscheidet der Vorstand auf seiner Sitzung zu Beginn des Jahres auch über den oder die Ehrenpreisträger:in. Schaut man zurück, so ist der Kreis der seit 2010 ausgezeichneten Personen und Gruppen ein Querschnitt durch die vielfältige TEX-Landschaft (`https://www.dante.de/dante-e-v/ehrenmitglieder-und-preise/`).

Besonders freut mich, dass alle Preisträger:innen auch heute noch sehr aktiv sind und ihre Preiswürdigkeit dadurch fortlaufend bestätigen.

In diesem Jahr zeichnen wir wieder ein Team aus. Wir gratulieren dem Fachgebiet Didaktik der Informatik der Bergischen Universität Wuppertal unter der Leitung von Ludger Humbert ganz herzlich zum Ehrenpreis 2022 von DANTE e.V. und danken dem gesamten Team für seinen lang anhaltenden Einsatz und seinen klaren Kurs. Die Gruppe setzt sich seit vielen Jahren vehement für den Einsatz von LATEX und Open-Source-Software in einem Bereich ein, der viele Herausforderungen mit sich bringt: die Lehrerausbildung. Verbunden mit der Auszeichnung ist ein gestiftetes Preisgeld von 1000 € und die herzliche Einladung zu einer unserer nächsten Tagungen.

Am Schluss habe ich die traurige Pflicht, an diejenigen Mitglieder zu erinnern, die im vergangenen Jahr von uns gegangen sind. Wir gedenken den Verstorbenen Walter Schmidt (2224), Uwe Plog (3778), Roland Wild (4447), Jens Börstinghaus (4867), Meinolf Liedhegener (5043), Patrick Happel (5054) und Ingo Schütze (6644).

Ich weiß, dass gerade der Tod von Walter Schmidt viele Mitglieder betroffen gemacht hat. Umso tröstender ist es, dass es in dieser Ausgabe der Mitgliederzeitung einen Nachruf auf ihn gibt. Ich danke dem TEX-Stammtisch Erlangen ganz herzlich für dieses besondere Andenken. Auch das sind Beiträge in Die TEXnische Komödie, die ich persönlich für sehr wichtig erachte. Daher kann ich Sie nur immer wieder ermuntern, sich auch in diesem Bereich einzubringen, damit wir als Verein an unsere Mitglieder angemessen erinnern können.

In diesem Sinne viel Vergnügen bei der weiteren Lektüre.

Herzlichst Ihr/Euer
Martin Sievers

Einladung zur Frühjahrstagung 2022 und 64. Mitgliederversammlung von DANTE e.V. an der Otto-von-Guericke-Universität Magdeburg

Martin Sievers, Mathias Magdowski

Liebe Mitglieder von DANTE e.V. ,

wir laden Sie ganz herzlich zur Frühjahrstagung 2022 vom 23.–25. Juni 2022 in Magdeburg ein.

Es kann aktuell noch nicht abgeschätzt werden, ob und ggf. welche coronabedingten Einschränkungen herrschen werden. Wir sind aber guter Dinge, dass die Verlegung in den Frühsommer eine Veranstaltung vor Ort möglich macht. Gleichzeitig behalten wir die Lage im Auge und planen auch mit einer digitalen Teilnahme im Sinne einer Hybridveranstaltung.

Der Zeitplan sieht wie folgt aus:

Mittwoch, 22. Juni,	ab 19 Uhr:	Vorabendtreff
Donnerstag, 23. Juni,	9 bis 17 Uhr:	Tutorien und Vorträge
	ab 19 Uhr:	Abendtreff
Freitag, 24. Juni,	9 bis 17 Uhr:	Vorträge
	ab 19 Uhr:	Tagungsessen
Samstag, 25. Juni,	ab 9 Uhr:	64. Mitgliederversammlung
	anschließend	Touristikprogramm (geplant)

Die Mitgliederversammlung beginnt am Samstag, den 25. Juni 2022, um 9 Uhr im

Guericke-Zentrum
Schleinufer 1
39104 Magdeburg
http://www.ovgg.ovgu.de/Erleben/Guericke_Zentrum-p-82.html

Der genaue Raum wird rechtzeitig bekannt gegeben und vor Ort ausgeschildert. Auch für die Mitgliederversammlung gilt, dass sie ggf. coronabedingten Einschränkungen vor Ort unterliegt. In diesem Fall findet die Mitgliederversammlung je nach Lage zusätzlich oder auch ausschließlich in einem virtuellen Konferenzraum statt.

Die Tagesordnung lautet:

1. Begrüßung und Tagesordnung
2. Bericht des Vorstands
3. Finanzbericht
4. Bericht der Rechnungsprüfer
5. Entlastung des Vorstands für 2020
6. Entlastung des Vorstands für 2021
7. Wahl von Rechnungsprüfern
8. Antrag des Vorstands auf Satzungsänderung: Ergänzung von § 12 Abs. (1)
9. Verschiedenes

Begründung des Antrags auf Satzungsänderung

§ 12 Abs. (1) lautet bisher:

> Mindestens einmal jährlich findet eine Mitgliederversammlung statt; sie soll in Verbindung mit einer Fachtagung abgehalten werden.

Die Entwicklungen seit Beginn der Coronapandemie haben deutlich gemacht, dass »hybride« oder auch »virtuelle« Mitgliederversammlungen sehr positiv aufgenommen werden. Für die Zeit nach Auslaufen der aktuell bis zum 31. 8. 2022 gültigen Sonderregelungen zur Durchführung »virtueller« Mitgliederversammlungen auch ohne ausdrückliche Regelung in der Vereinssatzung will der Vorstand eine dauerhafte Möglichkeit dazu schaffen. Grundsätzlich bleibt es aber bei der Mitgliederversammlung in Präsenz als »Normalfall«, die weiterhin in Verbindung mit einer Fachtagung abgehalten werden soll.

Der Vorstand bittet daher die Mitgliederversammlung, § 12 Abs. (1) wie folgt zu ergänzen (neuer Wortlaut hervorgehoben):

> (1) Mindestens einmal jährlich findet eine Mitgliederversammlung statt; sie soll *grundsätzlich* in Verbindung mit einer Fachtagung *in Anwesenheit der Teilnehmenden vor Ort abgehalten werden. Der Vorstand kann vorsehen, dass Mitglieder auch ohne Anwesenheit am Versammlungsort an der Mitgliederversammlung teilnehmen und ihre Mitgliederrechte im Wege der elektronischen Kommunikation ausüben können oder müssen.*

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. Wie üblich sind auch Nichtmitglieder als Gäste herzlich willkommen.

Unter `https://www.dante.de/veranstaltungen/dante2022/` finden Sie die Tagungsseite mit allen weiteren Informationen rund um die Veranstaltung. Neuigkeiten zur Tagung gibt es über die Vereinsmailingliste sowie über Twitter (`@dante_ev`). Für

alle Nachrichten zur Veranstaltung sollte dabei der Hashtag #DANTE2022 verwendet werden, gerne in Verbindung mit #TeXLaTeX.

Wir bitten diesmal noch dringender als sonst wegen der besseren Planbarkeit um eine frühzeitige Anmeldung über die Tagungswebsite.

Für Fragen, Wünsche und Anregungen schreiben Sie bitte an dante2022@dante.de oder wenden Sie sich postalisch an

DANTE e.V.
Stichwort: DANTE 2022
Postfach 11 03 61
69072 Heidelberg

Beiträge gesucht (»Call for Presentations«)

Wir möchten als Organisatoren natürlich ein spannendes und vielfältiges Vortragsprogramm anbieten können. Dazu sind wir allerdings auf Eure/Ihre 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,
- Einführungen in TEX und die zugehörigen Makropakete (LATEX 2_ε/LATEX3, ConTEXt, …),
- Lösungen mit den neueren Engines XETEX und LuaTEX,
- 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,
- barrierefreie PDF-Dokumente,
- …

Fühlen Sie sich angesprochen? Dann senden Sie bitte eine E-Mail mit folgenden Angaben an dante2022@dante.de:

– Name der Referentin/des Referenten,
– Titel und Art (Vortrag oder Tutorium) des Beitrags,
– 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.),
– kurze Zusammenfassung (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.

Mit freundlichen Grüßen
Martin Sievers (Vorsitzender DANTE e.V.)
Mathias Magdowski (OVGU Magdeburg)

Kombinierte Mitgliedschaft

Doris Behrendt

Auch dieses Jahr gibt es wieder die Möglichkeit einer kombinierten Mitgliedschaft bei DANTE e.V. und der TEX Users Group. Im Vergleich zum letzten Jahr ändert sich wenig, lediglich der Aufschlag für die TUG fällt aufgrund des für uns ungünstigeren Wechselkurses diesmal geringfügig höher aus.

Die Mitgliedschaft bei der TUG gilt jeweils für ein Kalenderjahr. Im Gegensatz zur Mitgliedschaft bei DANTE e.V. verlängert sie sich nicht automatisch. Sie können (bzw. müssen) sich also jedes Jahr neu entscheiden, ob Sie zusätzlich Mitglied der TUG sein möchten.
Die folgenden Beträge müssen zusätzlich zum Mitgliedsbeitrag bei DANTE e.V. entrichtet werden!

- 57,– € für reguläre Privatmitglieder sowie
- 39,– € für ermäßigte Privatmitglieder (Schüler, Studenten, Rentner etc.).

Für die sogenannte »Electronic Option« (Zeitschrift *TUGboat* nur als PDF zum Herunterladen) sind es

- 44,– € bzw.
- 27,– € ermäßigt (Schüler, Studenten, Rentner etc.).

Wenn Sie als Mitglied von DANTE e.V. zusätzlich Mitglied der TUG werden wollen, schreiben Sie bitte eine E-Mail an das Büro (`office@dante.de`) und überweisen den

oben genannten Zusatzbeitrag auf eines unserer Konten. Alternativ kann das Büro einen Lastschrifteinzug vornehmen oder Sie bezahlen per PayPal.

Sie gelten als reguläres Mitglied der TUG und erhalten drei Ausgaben des *TUGboat* im Jahr (jeweils den Jahrgang, auf den sich die kombinierte Mitgliedschaft bezieht). Gegenüber der direkten Mitgliedschaft bei der TUG bestehen jedoch zwei Unterschiede:

- Kombinierte Mitglieder erhalten von der TUG kein kostenfreies Exemplar der »TEX-Collection«, da diese bereits automatisch oder gegen einen geringen Beitrag für Mitglieder von DANTE e.V. erhältlich ist.
- Kombinierte Mitglieder erhalten von der TUG keine kostenfreie Zusendung gemeinsamer Veröffentlichungen von DANTE e.V. und TUG.

Nachruf: Walter Schmidt (*1960 – † 2021)

TEX-Stammtisch Erlangen

Walter Schmidt schied am 12.10.2021 freiwillig aus dem Leben. Der Erlanger TEX-Stammtisch trauert um einen seiner Gründer und zuverlässigsten Besucher und um einen gemeinsamen Freund.

Mit Walter haben wir einen vielseitig interessierten Menschen, einen geschätzten Gesprächspartner und einen guten Freund verloren. Er war die treibende Kraft unseres Stammtisches über mittlerweile Jahrzehnte, die Konstante der Organisation und ganz nebenbei natürlich auch beim jährlichen Treffen der Bayerischen TEX-Stammtische, den Weihnachtsfeiern und der Pflege von Netzwerken in der Szene engagiert. Kam eine TEXnische Frage auf, hatte Walter immer etwas zu sagen – durch seine lange Erfahrung, großes Wissen und seine Hartnäckigkeit manche harte Nuss knacken können. Auch bei unzähligen anderen Themen war Walter eine wandelnde Enzyklopädie, allem gegenüber aufgeschlossen und für viele Dinge zu begeistern.

Legendär ist das Sortiment an obskuren Musikgruppen und Fernsehserien, in denen er bewandert war. Viele Diskussionen am TEX-Stammtisch gingen im wörtlichen Sinne um Gott und die Welt, ersterer kompetent vertreten durch unsere TheologInnen, zweitere auch vom naturwissenschaftlich geprägten Publikum einer Universitätsstadt – Walter war immer ein geschätzter und interessierter Diskussionspartner.

Als DJ und Moderator beim Internet-Sender »Radio Stone.fm« brachte Walter viele von uns in Kontakt mit vormals ungehörten musikalischen Schätzen. Sein Faible für die Science-Fiction-Serie »Raumschiff Orion« war der Grund für seine nicht

einfach zu merkende Web-Adresse, der Name war das Rufzeichen von Commander McLane aus dieser Serie.

Walter war für Technik in allen Schattierungen zu haben, geradezu besessen beim Thema Eisenbahnen und Garant für Abschweifungen jenseits der Welt von TEX und Typographie, für die wir den Erlanger Stammtisch schätzen.

Mit technischem Verständnis und einem Talent für spannendes Erzählen unterhielt er uns oft mit seinen Schilderungen. Sehr lebhaft in Erinnerung blieb uns, wie er die gravierenden architektonischen Mängel eines Parkhauses erläuterte, veranschaulicht mit Salz- und Pfefferstreuern, was für schallendes Gelächter in unserer Runde sorgte.

So manche von uns hatte Walter in den ersten Schritten mit TEX und in komplizierteren Anwendungen von TEX begleitet. Egal ob es um Kennzeichnungen von geschlechtsneutralen Formulierungen oder um theologisch-linguistische Feinheiten ging, Walter unterstützte uns bei der Suche nach einer Lösung in TEX und LATEX. Er half vielen Leuten, auch außerhalb des Stammtisches, bei TEX-Sorgen und war selbstverständlich zur Stelle, wenn es um Märchenbücher, Briefköpfe und Diplomarbeiten ging.

Mit seiner wendigen, akzentuierten Stimme erklärte er typografische Spezialitäten und deren historischen Hintergründe. Detailverliebt muss man ihn nennen, und das ist ein Kompliment – er achtete auf Kleinigkeiten, Buchstaben, Ligaturen, Abstände und Linien. Seine Meinung zu optischem Randausgleich und zum »Augenpulver« (zu klein gesetzter Text ohne sichtbare Strukturierung) sind unvergesslich. Er achtete auf so vieles, was andere schlichtweg übersehen. Und so verrückt es klingt: Das ist auch der Grund, warum wir als der bunte Haufen, der wir sind, immer wieder zusammensitzen. Er war einer der treibenden Neugierigen, der über den eigenen Tellerrand hinweg Fragen stellte und Zusammenhänge suchte. Er hatte einen feinen Sinn für intellektuelle Scherze und nicht selten schaukelte sich der Resonanzraum in launige Stimmung.

Der kluge, neugierige und herzliche Mann, der die Musik und die Typographie so liebte – möge er jetzt mit der Raumpatrouille Orion am Rande der Unendlichkeit seine Ruhe finden.

Bretter, die die Welt bedeuten

Tabellen mit dem LaTeX-Paket tabularray

Rolf Niepraschk

Einleitung

Bereits in Standard-LaTeX gibt es die Möglichkeit zum Setzen von Tabellen. Die dort vorhandenen Schwächen und Defizite auszugleichen, haben sich viele LaTeX-Pakete zur Aufgabe gemacht. Beispiele sind das Paket `longtable` [1] zum Setzen von mehrseitigen Tabellen und das Paket `booktabs` [2] für spezielle Layoutverbesserungen. Aber auch Pakete, die vorrangig anderen Zwecken dienen, unterstützen gelegentlich das Setzen von Tabellen. So definiert beispielsweise das Paket `siunitx` [4] spezielle Spaltenparameter zum korrekten Setzen von Dezimalzahlen. In diesem Artikel soll das relativ neue LaTeX-Paket `tabularray` vorgestellt werden, das den Anspruch erhebt, eine Vielzahl von Verbesserungen hinsichtlich des Setzens von Tabellen in sich zu vereinen. Das Paket bietet eine große Anzahl von Konfigurationsmöglichkeiten, die hier nur angedeutet werden können. Es sei daher auf die umfangreiche Dokumentation verwiesen (siehe [3]). Im Folgenden wird an Hand von ausgewählten Beispielen die Verwendung des Paketes gezeigt.

In den Code-Ausschnitten wird das grundsätzlich nötige Laden des Paketes `tabularray` per

```
\usepackage{tabularray}
```

nicht extra angegeben.

Mehrzeilige Tabellenfelder

Zeilenumbrüche innerhalb von Tabellenfeldern lassen sich mit dem Paket `tabularray` verblüffend einfach erreichen:

```
\begin{center}
\begin{tblr}{|l|c|r|} \hline
  {links \\ l} & {mittig \\ m} & {rechts \\ r} \\ \hline
\end{tblr}
```

```
\end{center}
```

links l	mittig m	rechts r

Zeilen- und Spaltenzähler

Manchmal ist es nützlich, innerhalb von Tabellenfeldern zu wissen, in welcher Zeile und Spalte man sich gerade befindet. tabularray bietet dafür die Zähler rownum und colnum sowie für die Gesamtanzahl rowcount und colcount.

Wegen der besseren Übersichtlichkeit wurde vorher

```
\newcommand*\Spalte{Spalte:~\arabic{colnum}}
```

definiert.

```
\begin{center}
\begin{tblr}{colspec={r}, column{1}={preto=\arabic{rownum}.}, hlines, vlines}
  &  Gesamtzeilenzahl:~\arabic{rowcount} &
    Gesamtspaltenzahl:~\arabic{colcount} & \Spalte & \Spalte \\
  & \Spalte & \Spalte & \Spalte & \Spalte \\
  & \Spalte & \Spalte & \Spalte & \Spalte
\end{tblr}
\end{center}
```

1.	Gesamtzeilenzahl: 3	Gesamtspaltenzahl: 5	Spalte: 4	Spalte: 5
2.	Spalte: 2	Spalte: 3	Spalte: 4	Spalte: 5
3.	Spalte: 2	Spalte: 3	Spalte: 4	Spalte: 5

Ausrichtung von Dezimalzahlen am Dezimaltrenner

Zum Aktivieren sind zusätzlich die Anweisungen

```
\UseTblrLibrary{siunitx}
\sisetup{locale=DE}
```

in der Dokumentpräambel anzugeben.

```
\begin{tblr}{hlines, vlines, columns={6em},
  colspec={Q[si={table-format=3.2,table-number-alignment=left},l]
           Q[si={table-format=3.2,table-number-alignment=center},c]
           Q[si={table-format=3.2,table-number-alignment=right},r]}}
  {{{links}}} & {{{mittig}}} & {{{rechts}}} \\
  111         & 222          & 333          \\
```

```
  1.1         & 2.2          & 3.3          \\
  11.11       & 22.22        & 33.33        \\
\end{tblr}
```

links	mittig	rechts
111	222	333
1,1	2,2	3,3
11,11	22,22	33,33

Mehrseitige Tabelle mit Spaltenparameter »X«

Der Spaltenparameter »X« war bisher schon von dem Paket tabularx bekannt. Er definiert eine zur p-Spalte ähnliche Spalte, deren Breite automatisch so berechnet wird, dass sie den gesamten noch zur Verfügung stehenden horizontalen Platz einnimmt. Sowohl die Standard-Tabellenumgebung tblr[1] in tabularray, als auch longtblr, das Äquivalent zur Umgebung longtable des gleichnamigen Paketes, kennen diesen nützlichen Spaltenparameter.

Um im folgenden Beispiel die Fortsetzungshinweise in deutscher Sprache zur erhalten, wurde

```
\DefTblrTemplate{contfoot-text}{default}{Fortsetzung auf der nächsten Seite}
\DefTblrTemplate{conthead-text}{default}{(Fortsetzung)}
```

definiert.

```
\begin{longtblr}[caption={Lange Tabelle}]{|c|>{\dotfill}X<{\dotfill}|c|}\hline
  Alpha & Beta & Gamma \\ \hline
  ...
  Alpha & Beta & Gamma \\ \hline
\end{longtblr}
```

Table 1: Lange Tabelle

Alpha	.....Beta.....	Gamma
Alpha	.....Beta.....	Gamma
Alpha	.....Beta.....	Gamma

Fortsetzung auf der nächsten Seite

[1] Der Name ist ein witziges Synonym sowohl für ***t**a**b**u**l**a**r*** als auch für ***t***op, ***b***ottom, ***l***eft, ***r***ight.

Table 1: Lange Tabelle (Fortsetzung)

Alpha	.....Beta.....	Gamma
Alpha	.....Beta.....	Gamma
Alpha	.....Beta.....	Gamma
Alpha	.....Beta.....	Gamma
Alpha	.....Beta.....	Gamma
Alpha	.....Beta.....	Gamma

Normale Tabelle mit Spaltenparameter »X«

```
\begin{tblr}{|X|X[3,c]|X[2,r]|}\hline
  links & mittig & rechts \\ \hline
\end{tblr}
```

links	mittig	rechts

Tabellenlayout ähnlich dem des Paketes booktabs

Zum Aktivieren ist zusätzlich folgende Anweisung

```
\UseTblrLibrary{booktabs}
```

in der Dokumentpräambel anzugeben.

```
\begin{booktabs}{@{}llll@{}} \toprule
  Alpha   & Beta  & Gamma   & Delta \\ \midrule
  Epsilon & Zeta  & Eta     & Theta \\ \cmidrule[r]{1-3}
  Iota    & Kappa & Lambda  & Mu    \\ \cmidrule[l]{2-4}
  Nu      & Xi    & Omikron & Pi    \\ \bottomrule
\end{booktabs}
```

Alpha	Beta	Gamma	Delta
Epsilon	Zeta	Eta	Theta
Iota	Kappa	Lambda	Mu
Nu	Xi	Omikron	Pi

Gefärbte Tabellenzeilen

Voraussetzung für das Anfärben von Bereichen einer Tabelle ist das Laden des Paketes `xcolor`.

```
\begin{center}
\begin{tblr}{colspec={cccc}, row{odd}={bg=azure5}, row{even}={bg=green7},
  row{1}={fg=white,bg=purple3,font=\bfseries}}
  Spalte 1 & Spalte 2 & Spalte 3 & Spalte 4 \\
  Alpha    & Beta     & Gamma    & Delta    \\
  Epsilon  & Zeta     & Eta      & Theta    \\
  Iota     & Kappa    & Lambda   & Mu       \\
  Nu       & Xi       & Omikron  & Pi       \\
\end{tblr}
\end{center}
```

Spalte 1	Spalte 2	Spalte 3	Spalte 4
Alpha	Beta	Gamma	Delta
Epsilon	Zeta	Eta	Theta
Iota	Kappa	Lambda	Mu
Nu	Xi	Omikron	Pi

Gefärbte Tabellenspalten

```
\begin{tblr}{colspec={X[c]X[c]X[c]X[c]},
  column{odd} ={bg=green7}, column{even} ={bg=azure5},
  cell{1}{odd}={bg=green3}, cell{1}{even}={bg=azure1}, cells={font=\sffamily},
  row{1}={fg=white,font=\bfseries\sffamily}, hline{2-Y}={.35ex,white}}
  Spalte 1 & Spalte 2 & Spalte 3 & Spalte 4 \\
  Alpha    & Beta     & Gamma    & Delta    \\
  Epsilon  & Zeta     & Eta      & Theta    \\
  Iota     & Kappa    & Lambda   & Mu       \\
  Nu       & Xi       & Omikron  & Pi       \\
\end{tblr}
```

Spalte 1	Spalte 2	Spalte 3	Spalte 4
Alpha	Beta	Gamma	Delta
Epsilon	Zeta	Eta	Theta
Iota	Kappa	Lambda	Mu
Nu	Xi	Omikron	Pi

Spezielle Tabelleninhalte

Nicht alle Tabelleninhalte lassen sich problemlos in die Zellen einer Tabelle einfügen. Insbesondere »vertikales Material«, wie z. B. die `itemize`-Umgebung, widersetzt sich, horizontal ausgemessen zu werden, wie es aber für das korrekte Erstellen einer Tabelle notwendig ist. `tabularray` bietet als Lösung den Parameter `measure=vbox` an. Zum Aktivieren ist zusätzlich die Anweisung

```
\UseTblrLibrary{varwidth}
```

in der Dokumentpräambel anzugeben. Der verwendete generische Spaltenparameter »`Q`« gestattet Angaben zur horizontalen und vertikalen Ausrichtung.

```
\begin{tblr}{colspec={Q[c,m]Q[c,m]Q[c,m]}, hlines, vlines, measure=vbox}
  \begin{itemize} \item Nu \item Xi \item Omicron \end{itemize} &
  Kappa &
  \begin{itemize} \item Mu \item Pi \item Eta \end{itemize} \\
  Gamma & Beta &  Alpha
\end{tblr}
```

• Nu • Xi • Omicron	Kappa	• Mu • Pi • Eta
Gamma	Beta	Alpha

Literatur

[1] David Carlisle: The longtable package, Version v4.16, 2021, `CTAN:macros/latex/required/tools/longtable.pdf` (besucht am 1. 10. 2021).

[2] Simon Fear: Publication quality tables in LaTeX, Version v1.61803398, 2020, `CTAN:macros/latex/contrib/booktabs/booktabs.pdf` (besucht am 1. 10. 2021).

[3] Jianrui Lyu: Tabularray. Typeset Tabulars and Arrays with LaTeX3, Version 2021P, 2021, `CTAN:macros/latex/contrib/tabularray/tabularray.pdf` (besucht am 1. 10. 2021).

[4] Joseph Wright: siunitx – A comprehensive (si) units package, Version v3.0.32, 2021, `CTAN:macros/latex/contrib/siunitx/siunitx.pdf` (besucht am 1. 10. 2021).

Having Fun with LaTeX: Eine tolle Masche

Adelheid Bonnetsmüller

Früher war Stricken als etwas Altbackenes verschrieen, dann wurde es in der grünen, alternativen Szene zum großen Hit (wer erinnert sich nicht an die ersten Grünen im Bundestag, die strickend in den Reihen saßen?). Und heute? Heute ist Stricken wieder groß im Kommen und erfreut sich zunehmend bei Jung und Alt aus allen »Ecken« steigender Beliebtheit. Zeit also, um ein LaTeX-Paket vorzustellen, mit dem Stricken erklärt werden kann: Mit `knitting` lassen sich Strickschriften setzen.

Die Serie »Having Fun with LaTeX« widmet sich den kleineren Paketen auf CTAN und stellt einige vor. Oft schlummern solche Pakete in den dortigen großen Weiten, was schade ist. Nicht selten können sie einem das LaTeX-Leben leichter, bunter und schöner machen. Oder sie bringen einen zum Schmunzeln. Oder sind eben für manche Anwender das, was sie schon länger gesucht haben – und nicht damit gerechnet haben, dass ein anderer hierfür ein Paket geschrieben haben könnte. »Having Fun with LaTeX« gibt es in Form von Vorträgen oder Beiträgen in Die TeXnische Komödie seit der Frühjahrstagung von DANTE e.V. 2009 in Wien und wird nun nach einigen »stillen« Jahren fortgesetzt.

Generelles

Für Strickschriften gibt es neben `knittingpattern` das Paket `knitting` von Ariel Barton in der aktuellen Version von 2019. Dieser Artikel beschäftigt sich mit dem zweiten Paket, da es umfassender und mächtiger ist. Geladen wird es wie üblich mittels `\usepackage[Option]{knitting}`. Das Paket mit den zugehörigen Fontdateien ist Teil der TeXLive-Distribution. Sollte es nicht installiert sein, bitte den entsprechenden Anweisungen in der Paketdokumentation folgen.

Strickschrift ist ja nun für viele doch eher kryptisch – und leider auch nicht wirklich einheitlich in den gängigen Magazinen und Büchern. Allerdings ist in (fast) allen Strickmustern erklärt, welches Symbol für was steht. Die Bedeutung der im Paket zur Verfügung gestellten Symbole finden sich in der Datei `knitkey.pdf`, die im Dokumentationsordner des Pakets enthalten ist. Die Begriffe dort sind auf englisch, was in diesem speziellen Fall selbst bei sehr guten Englischkenntnissen eine Herausforderung darstellen kann. Als wesentlich sei hier genannt, dass *knit* eine rechte Masche und *purl* eine linke Masche bedeutet. Erstere werden in diesem Paket mit dem Symbol | auf weißem Grund, zweitere mit dem Symbol — auf grauem Grund dargestellt – oder eben nur mit einem leeren weißen bzw grauen Kästchen .

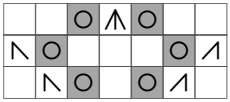
Als Option beim Laden des Pakets steht `chartsonly` zur Verfügung. Diese erzeugt für jedes Strickmuster eine Seite in einer PDF-Datei, die dann in andere Dateien eingebunden werden kann. `chartsonly` kann allerdings nur mit `latex` kompiliert werden, andernfalls wird ein Fehler ausgegeben.

Wolle und Nadeln in die Hand – und los geht's!

Ein paar erste einfache Maschen

Das Paket stellt den Befehl `\chart` zur Verfügung, in dessen Argument die Strickanleitung im Detail gepackt wird. Ein erstes Beispiel sieht folgendermaßen aus.

```
\chart{
--0A0--
<0---0>
-<0-0>-
}
```



Man erkennt, dass die einzelnen Strickanweisungen über kleine und große Buchstaben sowie einige Sonderzeichen gesteuert werden. Die oberste Zeile bedeutet hier beispielsweise: 1 Masche rechts, 1 Masche rechts, 1 linke Masche mit Garnüberzug[1], 1 Masche rechts auslassen, 2 Maschen rechts zusammenstricken und ausgelassene Masche drüberziehen, 1 linke Masche mit Garnüberzug, 1 Masche rechts, 1 Masche rechts.

Stricken im Fließtext

Möchte man in einem Fließtext, beispielsweise zur Erläuterung, ein Stricksymbol einfügen, geschieht dies über `\textknit`, hierbei müssen die üblich verdächtigen Zeichen mit einem \ geschützt werden, so ergibt z. B. `\textknit{\&}` Folgendes: V. (Innerhalb des Befehls \chart können diese problematischen Zeichen aber ohne weiteres verwendet werden.)

Verschiedene Wollfarben…

Für fast alle möglichen Symbole hat der Autor eine rechte und linke Variante entworfen. Diese werden unterschiedlich angesteuert. So erzeugt `\textknit{f}` das Symbol für »rechts abstricken« ⌒ und `\textknit{F}` ⌒ das für »links abstricken«[2].

[1] Das bedeutet, dass das Garn über die rechte Stricknadel geführt und die Masche links gestrickt wird. Dadurch entsteht eine zusätzliche Masche, die in der nächsten Reihe entweder mitgestrickt oder zusammengestrickt wird. So entstehen gewollte Löcher oder eben ein breiteres Strickstück.

[2] Abstricken bedeutet »endlich fertig«, vermutlich steht das »f« für »finish«.

Häufig werden für rechte Maschen kleine und für linke große Buchstaben verwendet, aber nicht immer.

Da zum einen für ein paar wenige Symbole nur die rechte oder linke Variante vordefiniert wurde und man es sich zum anderen manchmal einfacher machen möchte und nicht das Kürzel für den jeweils anderen Maschentyp raussuchen will, gibt es die Möglichkeit, dem Kürzel für eine rechte Masche den Hintergrund einer linken überzustülpen. Dies geschieht mittels dem Befehl `\purlbackground{Symbol der rechten Masche}`. So ergibt `\textknit{\purlbackground{f}}`  : der Hintergrund entspricht der einer linken Masche, das Symbolkürzel (`kleines f`) dem einer rechten Masche.

Die Hinter- und Vordergrundfarbe der linken Maschen lässt sich über das Neudefinieren der `\purlboxbg`- und `\purlboxfg`-Farbe ändern.

```
\definecolor{purlboxbg}{gray}{0.57}
\definecolor{purlboxfg}{gray}{0.2}
\renewcommand\purlboxbackground{\color{purlboxbg}}
\renewcommand\purlboxforeground{\color{purlboxfg}}
\textknit{AO\Purl[-1]{12}{6}}
```

Das –1 im optionalen Argument von `\Purl` sorgt im Übrigen dafür, dass die Zahl 12 ein Kästchen links von der Mitte gesetzt wird. Im Gegensatz dazu sieht man im Beispiel im Abschnitt »Abkürzungen«, dass die 12 genau mittig in Masche 3 **und** 4 gesetzt wird.

Auch andere Farben lassen sich steuern, die nachfolgende Tabelle liefert einen Überblick.

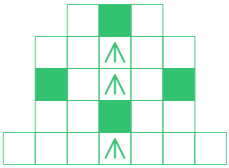
`gridcolor`	Farbe des Kästchenrahmens
`knitlinecolor`	Farbe der Linie beim Hervorheben von Mustern (siehe weiter hinten)
`rncolor`	Farbe der Zeilennummer (siehe weiter hinten)
`rnarrowcolor`	Farbe des kleinen Pfeils bei der Zeilennummer (siehe weiter hinten)

Zur Farbgebung muss man wissen, dass der Hinter- und Vordergrund getrennt voneinander gesteuert werden. Der Hintergrund ist beispielsweise in der Sequenz  weiß für die erste Masche und grau für die zweite. Die Vordergrundfarbe ist die des Kreises in der zweiten Masche, die Gitterfarbe die des Kästchengitters beider Maschen, also in beiden Fällen schwarz (das ist der Standard).

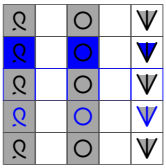
Möchte man nun einen farbigen Hintergrund haben, reicht es nicht aus, dem Befehl `\chart` oder `\textknit` nur eine Farbe mitzugeben, viel mehr muss den verschiedenen Elementen der Symbole über die Befehle `\purlpass`, `\gridpass`, `\mainpass` eine Farbe mitgegeben werden. Der erste Befehl ändert hierbei die Hintergrundfarbe der linken Maschen, der zweite die des Kästchenrahmens und der dritte die Farbe der

Symbole. Berücksichtigt man dies nicht, wird die angegebene Farbe wie im folgenden Beispiel sowohl dem Hinter-, dem Vordergrund wie auch dem Kästchenrahmen mitgegeben, was offensichtlich nicht zielführend ist.

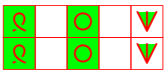
```
Falsches Vorgehen:
(O nicht mehr sichtbar
in 1. und 4. Reihe)
\chart{\color{blue!30!green!80}
~~-O-~~
~--A--~
~=-A-=~
~--O--
---A---
}
```



```
Richtiges Vorgehen:
\chart{
T-O-E
{\purlpass{\color{blue}}T-O-E}
{\gridpass{\color{blue}}T-O-E}
{\mainpass{\color{blue}}T-O-E}
T-O-E
}
```



```
oder globale Farbänderung mit
Änderung Hintergrundfarbe:
\chart{\color{red}
\purlpass{\color{green}}{
T-O-E
T-O-E}
}
```



Im ersten Beispiel wird zusätzlich die Verwendung der Tilde gezeigt. Diese symbolisiert, dass an der entsprechenden Stelle keine Masche vorhanden ist. Dies passiert, wenn Maschen zusammengestrickt werden (ob absichtlich oder unabsichtlich) oder wenn erst im späteren Verlauf des Strickwerks weitere Maschen aufgenommen werden.

...und verschiedene Wollarten (Schriftarten)

Das Paket stellt vier verschiedene »Schriftarten« für die Darstellung der Symbole zur Verfügung: `\knitgrid`, `\knitnogrid`, `\knitmixed` und `\knitwide`.

```
\knitgrid
\textknit{A=4x-}
```

Standardeinstellung, rechte und linke Maschen als graues bzw. weißes Kästchen dargestellt.

```
\knitnogrid
\textknit{A=4X-}
```

kein Rahmen um die Symbole.

```
\knitmixed
\textknit{A=4x-}
```

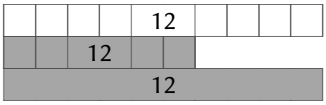
rechte und linke Maschen zusätzlich mit Symbol dargestellt.

```
\knitwide
\textknit{A=4X-}
```

breitere Kästchen, sinnvoll für komplexere Muster.

In der Kürze liegt die Würze

Beim Musterstricken ist es oft üblich, dass breite Stücke glatt rechts oder links gestrickt werden und zwischendurch evtl. ein Zopfmuster kommt. Sollen so beispielsweise 20 Maschen rechts gestrickt werden, kann man die Darstellung abkürzen, indem man den Befehl `\Knit{}{}` bzw. `\Purl{}{}` verwendet. Im ersten Argument wird die Anzahl der zu strickenden Maschen (im Beispiel 20) angegeben, im zweiten wieviele Kästchen beispielhaft dargestellt werden sollen. Mit dem Befehl `\knitbox {Zahl}{Länge der Box}` oder `\purlbox {Zahl}{Länge der Box}` werden hierbei keine einzelnen Kästchen gezeichnet, sondern eine Box mit der angegebenen Länge.

```
\chart{
\Knit{12}{10}
\Purl{12}{6}
\purlbox{12}{10}
}
```


Fehlende Maschen

Es kommt vor, dass durch vorheriges Zusammenstricken in der Folgereihe eine Masche weniger vorhanden ist, die aber gleich in der nächsten Reihe wieder aufgenommen wird. Um nicht das Gesamtbild der Strickschrift zu stören, gibt es die Möglichkeit diese fehlende Masche auszuweisen: ein `\textknit{.}` erzeugt das passende Symbol dazu ■.

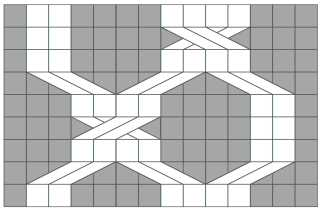
Kompliziertere Maschen

Für kompliziertere Strickanleitungen, z. B. beim Zopfstricken, bei dem Maschen auf einer Extranadel ausgelagert werden, um die Reihenfolge, in welcher dieser abgestrickt werden, zu ändern, braucht es einen Extracode. Diese Maschen sind

scheinbar sogar so kompliziert, dass es mir nicht möglich war, sie innerhalb der Dokumentenklasse für »Die TEXnische Komödie« zu setzen. Da ich den Fehler nicht gefunden habe, wurden die Beispiele für diesen Abschnitt in einem separaten Dokument mit Hilfe der Paketoption `chartsonly` gesetzt und anschliessend hier eingebettet. Ich erwähne das hier explizit, da dieses Problem vielleicht auch bei anderen Dokumentklassen auftritt. Es wird dabei kein Fehler ausgegeben, sondern ein schraffiertes Kästchen .

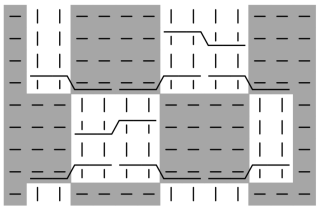
Beim Stricken von Zöpfen wird eine bestimmte Anzahl Maschen auf eine Hilfsnadel gelegt, dann (in der Regel) die gleiche Anzahl Maschen regulär gestrickt, bevor die auf der Hilfsnadel (Zopfnadel) befindlichen Maschen ebenfalls gestrickt werden. Dieses Verdrehen der Maschen erzeugt dann einen Zopf. In der Strickschrift gibt es zwei Möglichkeiten, dies darzustellen: einmal bildlich als Zopf, zum anderen mit Hilfe einer Linie, die über den zu verdrehenden Bereich von oben nach unten gezeichnet wird.

```
\chart{
=--====---===
=--====CCkk===
=--====---===
=CCppggKKCCpp=
===----=====--=
===ccKK====--=
===----=====--=
=ggKKCCppggKK=
=--====---===
}
```



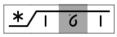
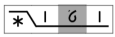
Die zweite Darstellungsform ergibt sich durch das Verwenden der »Schriftart« `knitnogrid` und sieht wie folgt aus.

```
\knitnogrid
\chart{
=--====---===
=--====CCkk===
=--====---===
=CCppggKKCCpp=
===----=====--=
===ccKK====--=
===----=====--=
=ggKKCCppggKK=
=--====---===
}
```



Da naturgemäß die Elemente in einem Zopf nicht nur aus rechten oder linken Maschen, sondern aus allen möglichen Kombinationen von Maschen bestehen können, erlaubt es das Paket, die Zopfsymbole der »Schriftart« `knitnogrid` mit anderen Symbolen zu kombinieren. Für die Variante der bildlichen Darstellung von Zöpfen stehen 12 verschiedene Möglichkeiten zur Verfügung (Unterschied in Breite und Farbgebung).

Es existieren des weiteren die Befehle `\cableright` und `\cableleft`, deren Verwendung folgendes Beispiel erläutert.

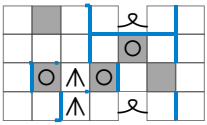
`\textknit{\cableright{*}{-Q-}}`	rechter Zopf mit weiteren Anweisungen[3]	
`\textknit{\cableleft{*}{-Q-}}`	linker Zopf mit weiteren Anweisungen[3].	

Muster hervorheben

Oft sieht man in Strickschriften, dass bestimmte Muster, die durch eine immer gleich bleibende Abfolge von Maschen entstehen, im Laufe des Strickwerks wiederholt werden sollen. Um in einem größeren Abschnitt ein solches Muster hervorzuheben oder beispielsweise für eine bestimmte Sequenz eine Abkürzung einzuführen, können solche Musterabschnitte mit einer *Outline* hervorgehoben werden. Hierbei erzeugt `\!` eine senkrechte Linie in der gewählten Farbe und `\_` eine waagrechte. `\overline{Maschenabfolge}` erzeugt eine Linie über die in den geschweiften Klammern definierten Maschen. Die Syntax ist im folgenden Beispiel erkenntlich. Falls die waagrechten Linien nicht über `\overline` erzeugt werden, sondern mit `\_`, ist zu beachten, dass die `\_` innerhalb einer »Pseudozeile« (`~~~~~`) erzeugt werden müssen.

[3] Der $*$ bedeutet, dass drei Maschen links (oder rechts) zusammengestrickt werden, aber nicht auf die rechte Nadel gehoben werden, sondern noch einmal rechts (bzw. links) und einmal links (bzw. rechts) gestrickt werden. Dann folgt eine rechte Masche, dann eine verschränkte linke Masche und wieder eine rechte Masche. Die letzten drei Maschen sind dabei Teil des Zopfes, d.h. es liegen dazwischen drei Maschen auf einer Zopfnadel, die erst nach diesen drei Maschen gestrickt werden.

```
\knitgrid
\definecolor{knitlinecolor}{rgb}{0,0.5,0.8}
\chart{
~~~\_\_\_~
-=-\!-U-\!-
---\overline*{\!-O-\!}-
~\_\_~\_\_~ \\ % eine Pseudozeile
-\!OAO\!-=-
~\_~~\_\_~ \\
--\!A-U-\!-
~~\_\_\_\_~
}
```



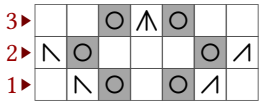
Reihen und Maschen zählen

Jeder, der schon mal gestrickt hat, weiß wie wichtig es ist, einen Überblick über die aktuelle Maschenzahl zu haben, vor allem, wenn des öfteren Maschen zusammengestrickt oder aufgenommen werden müssen (und wie übel es ist, wenn die Zahl nicht stimmt). Beim Musterstricken ist es fast genauso wichtig zu wissen, in welcher Reihe man sich befindet. Beides ist leicht steuerbar.

Erstmal wissen, in welcher Reihe man ist…

Über die Angabe von `left`, `right`, `both` sowie `evenleft`, `oddleft`,`evenright` und `evenleft` als optionales Argument beim Befehl `\chart` werden die entsprechenden Reihen rechts, links, beidseitig oder die geraden Zahlen auf der einen, die ungeraden auf der anderen Seite des Musters angezeigt. Hierbei wird jeweils der Wert des Zählers `rownumber` ausgegeben; dieser wird dann automatisch nach der Ausgabe um 1 erniedrigt. Ist der `\chart`-Befehl beendet, wird der Zähler wieder zurückgesetzt (`resetrn`).

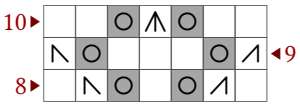
```
\chart[left]{
--OAO--
‹O---O›
-‹O-O›-}
```



Zu beachten ist, dass bei Strickmustern immer die unterste Reihe die erste Reihe ist, diese wird also folgerichtig mit der Zahl `1` gekennzeichnet. Soll dies nicht der Fall sein, und die unterste Reihe mit einer anderen Zahl gekennzeichnet werden, weil sich ein Muster beispielsweise über mehr als eine Seite erstreckt und man auf der zweiten Seite bereits mit einer höheren Zeilennummer beginnen möchte, kann dieser Mechanismus bei der Fortsetzung des Strickmusters mit `\resetrnfalse`

abgeschaltet werden (und mit `\resetrntrue` wieder angeschaltet werden), die Nummer der obersten(!) Reihe wird dann mittels `\setcounter{rownumber}{oberste(!) Zeilennummer}` auf einen anderen Wert gesetzt.

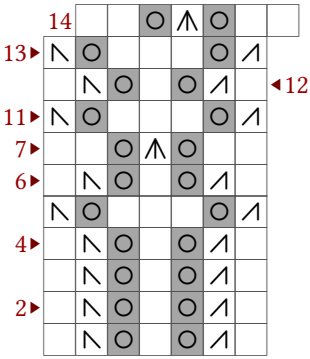
```
\resetrnfalse
\setcounter[evenleft]{rownumber}{10}
\chart{
--OAO--
<0---0>
-<0-0>-}
\resetrntrue
```



Möchte man in einem längeren Musterstück nur bestimmte Reihennummern ausgeben, so bietet es sich statt der Option für die generelle Zeilennummernangabe an, diese Reihen innerhalb des `chart`-Befehls mittels `rn` zu kennzeichnen. Zu beachten ist, dass das Verwenden von `\rn` die Zeilennummer ohne Pfeil ausgibt und die Darstellung der Reihe um ein Kästchen verschiebt.

Für die weitere Anpassung der Nummerierung innerhalb des `chart`-Befehls gibt es mehrere Befehle. `\rnright` gibt den Wert des Zählers `rownumber` rechts, `\rnleft` links von der aktuellen Reihe aus. Möchte man mehrere Zeilennummern überspringen, kann der Wert des Zählers mit dem Befehl `\addtocounter{rownumber}{-ZAHL}` angepasst werden. `\nonumber` unterdrückt die Ausgabe einer Zahl und mit `\rnoddonly` bzw. `\rnevenonly` werden nur ungerade (bzw. gerade) Zeilennummern ausgegeben (mit `\rnnormal` kann dieses Verhalten wieder auf den Standard zurückgesetzt werden).

```
\chart{
\rn --OAO--
\rnleft <0---0>
-<0-0>- \rnright
\rnleft <0---0>
\addtocounter{rownumber}{-3}
   \rnleft --OAO--
\rnleft -<0-0>-
\rnevenonly \rnleft<0---0>
\rnevenonly \rnleft-<0-0>-
\rnevenonly \rnleft-<0-0>-
\rnevenonly \rnleft-<0-0>-
\rnevenonly \rnleft-<0-0>-
}
\rnnormal
```

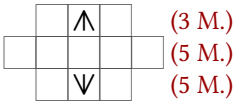


...und wieviel Maschen man haben sollte

Neben der Zeilennummer kann auch die Anzahl der Maschen in der jeweiligen Zeile ausgegeben werden. Hierfür steht der Befehl `\countstitches{}` zur Verfügung, in dessen Argument die Maschenkodierung der jeweiligen Reihe steht. Der Befehl schreibt in den Zähler `stitchcountin` die Anzahl der Maschen **bevor** die Reihe begonnen wird und in den Zähler `stichcountout` die Anzahl der Maschen, **nachdem** die Reihe gestrickt wurde.

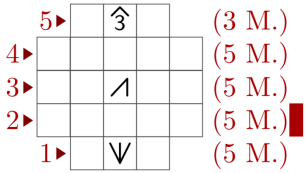
Um nun zu erreichen, dass dieser Zähler pro Reihe ausgegeben wird, definiert man sich am besten ein eigenes Kommando, bei dem man die Ausgabe auch gleich noch modifizieren kann. Ein erstes Beispiel zeigt die Ausgabe der Maschenanzahl nachdem eine Reihe gestrickt wurde.

```
\newcommand{\mystitchcount}[1]{
   \countstitches{#1}#1
   \mainpass{{\color{red}
   \textnormal{ (\thestitchcountout\ M.)}}}}}

\chart{
\mystitchcount{~-A-~}
\mystitchcount{-----}
\mystitchcount{~-w-~}}
```



Der Befehl kann sogar genutzt werden, um Plausibilitätsprüfungen durchzuführen. So ist hinter jedem Stricksymbol hinterlegt, wieviele Maschen hierfür benötigt werden. Werden für eine Reihe also mehr Maschen benötigt, wie in der vorherigen Reihe vorhanden sind, kann eine Warnung ausgegeben werden. Das erleichtert das Schreiben von Strickschriften ungemein, funktioniert aber nur wenn in der LaTeX-Dokumentenklasse die `draft`-Option angegeben wird. Das zeigt das folgende Beispiel.

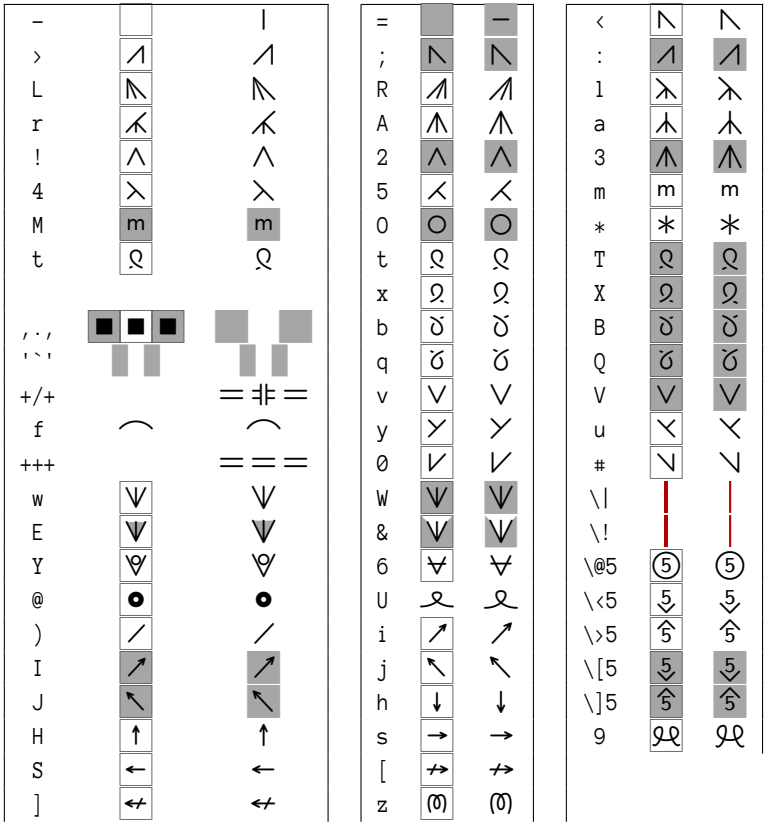
```
\newcommand{\mystitchcount}[1]{
   \countstitches{#1}#1
   \textnormal{\color{red}
   \mainpass{ (\thestitchcountout\ M.)
   \stitchcountwarningbar}}

\chart{
\setcounter{stitchcountin}{-100}
\mystitchcount{~-A-~}
\mystitchcount{-----}
\mystitchcount{--›--}
\mystitchcount{-----}
\mystitchcount{~-w-~}}
```



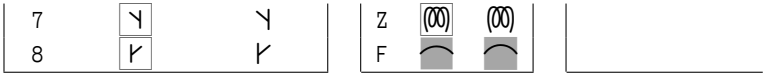
Das Setzen des Zählers `stitchcountin` auf `-100` hat hier den Hintergrund, dass ansonsten schon bei der ersten Reihe eine Warnung ausgegeben wird. Man erkennt, dass in der zweiten Reihe ein Warnbalken neben der Maschenanzahl ausgegeben wird, da durch die Maschenaufnahme in der ersten Reihe sechs Maschen auf der Nadel vorhanden sind und so auch sechs Maschen in der zweiten Reihe gestrickt werden müssen, angegeben sind in der Strickschrift aber nur fünf.

Viele, viele mögliche Maschen

In der folgenden Übersicht werden die (meisten[4]) möglichen Maschen (ohne Erklärung, wie diese zu stricken sind) gezeigt, jeweils in der »Schriftart« `\textgrid` und `\textnogrid`.



[4] Es gibt einige wenige Symbole, die in der `dtk`-Klasse nicht richtig dargestellt werden.



Eine neue Masche definieren

Wem das noch nicht reicht, der kann sich mit Hilfe des Befehls `\knitbox` weitere Symbole definieren, z. B. `\textknit{\knitbox{3 M auslagern}{6}}` **3 M auslagern**. Es können auch Symbole, z. B. aus der Mathematikschrift verwendet werden: `\knitbox{$\vspace{-1.5pt}\heartsuit$}{1}` wird zu ♡.

Erstellung eines Kalenders

Ralf Mispelhorn

Als Weihnachtsgeschenk habe ich einen Jahreskalender mit Bildern erstellt. Wichtige persönliche Tage wurden in dem Kalender durch farbige Kreise besonders gekennzeichnet.

Bilder konvertieren

Weil die Orginalbilder alle im `jpeg`-Format vorlagen, wurden sie mit Hilfe eines Shell-Skripts in das `png`-Format konvertiert. In einer Schleife werden alle `*.jpg` bzw. `*.jpeg` erfasst. Die Dateierweiterung wurde dann durch `.png` ersetzt und mit dem `convert`-Kommando von `imagemagick` die Konvertierung durchgeführt. Dabei wurde bei jedem Bild vorher geprüft, ob die Zieldatei bereits vorhanden ist. In diesem Fall ist keine Konvertierung durchgeführt worden.

```
1 for i in *.jp*g
2 do
3   S=$i
4   D=${i%.jp*g}.png
5   [ ! -f $D ] && convert $S $D
6 done
```

Kalender

Der Kalender besteht aus 12 Blättern, für jeden Monat ein Blatt. In diesem Artikel wird aus Platzgründen nur das erste Blatt für den Januar gezeigt.

Für die Generierung des Kalenders wurde das Paket `tikz` in der üblichen Form mit `\usepackage{tikz}` eingebunden und zusätzlich die `tikz`-Bibliothek `calendar` mit `\usetikzlibrary{calendar}` integriert.

Das Seitenlayout wurde mit `\pagestyle{empty}` eingestellt, damit keine Kopf- und Fusszeilen erscheinen.

Um Probleme mit Dateinamen zu verhindern, wurde zusätzlich das Paket `grffile` verwendet.

Deutsche Monatsnamen wurden mit Hilfe des `translator`-Paket erreicht.

Damit der Kalender einfach auf ein neues Jahr angepasst werden kann, wurde noch das Makro `\Year` eingeführt.

```
\documentclass[12pt,ngerman]{scrartcl}
\usepackage{babel}
\usepackage{translator}
\usepackage{grffile}
\usepackage{tikz}
\usetikzlibrary{calendar}
\pagestyle{empty}
\begin{document}

\begin{center}

\newcommand\Year{2022}
```

Als nächstes folgen mehrere Stil-Definitionen für die `tikz`-Umgebung.

`cal/.style` definiert eine Wochenliste mit dem Monatsnamen zentriert darüber.

Die weiteren Stildefinitionen sind für verschiedene Gruppen von Ereignissen. Wichtig ist hierbei `opacity=` um die Zahl des Datums durchscheinen zu lassen.

```
\tikzset{%
cal/.style={week list,month label above centered},%
family/.style={cyan,fill,thick,fill opacity=0.2},%
enkel/.style={blue,fill,thick,fill opacity=0.2},%
geschwister/.style={green,fill,thick,fill opacity=0.2},%
tod/.style={black,fill,thick,fill opacity=0.4},%
}
```

Danach folgen die 12 Monate, wobei jeder Monat in gleicher Weise angegeben wurde.

Monat

Als erstes wurde ein Foto ausgegeben, welches einheitlich auf 40% der verfügbaren Texthöhe skaliert wurde, damit alle Fotos die gleiche Höhe haben.

Anschließend folgt ein `tikzpicture`, in dem der Kalender für einen Monat produziert wird. Der Kalender bekommt ein Label, damit auf ihn später referenziert werden kann, und einen Datumsbereich für den aktuellen Monat. Damit das Ende des Monats nicht für jeden Monat anders angegeben werden muss, wurde `last` als Tag für das Ende verwendet.

Ereignisse

Nun wurden noch die besonderen Tage von Ereignissen aufgeführt. Hierzu wurden Kreise auf einen bestimmten Tag mit einer Stildefinition der Ereignisgruppe gezeichnet.

```
% Januar
\includegraphics[height=0.4\textheight]{roth.png}
\begin{tikzpicture}[scale=3,transform shape,remember picture]
\calendar(1) [dates=\Year-1-01 to \Year-01-last,cal];
\draw[family]      (1-\Year-01-14) circle (8pt);  % Person1
\draw[enkel]       (1-\Year-01-19) circle (8pt);  % Person2
\draw[geschwister] (1-\Year-01-21) circle (8pt);  % Person3
\draw[tod]         (1-\Year-01-27) circle (8pt);  % Person4
\end{tikzpicture}
\clearpage

% Februar
% ...
```

Fertiger Kalender

Januar

					1	2
3	4	5	6	7	8	9
10	11	12	13	14	15	16
17	18	19	20	21	22	23
24	25	26	27	28	29	30
31						

Abb. 1: Das Bild wurde im Jahr 2021 während eines Kurzurlaubs am Rothsee (Fränkisches Seenland) aufgenommen.

Lokale Seitenzähler innerhalb eines Dokuments

Tobias Hilbricht

Dieser Artikel erläutert, wie man ein Dokument mit lokalen Seitenzählern ausstatten kann, so dass mehrere Seitenbereiche mit voneinander unabhängigen Nummerierungen nach Art „Seite x von y“ möglich sind. Anschließend wird die Verwendung dieser Lösung im „LaTeX-Frontend“ LyX erläutert.

Einsatzzweck für mehrere lokale Seitenzähler in einem Dokument

Im Laufe der Jahre, in denen ich die Fächer Biologie und Chemie unterrichte, haben sich im Ordner „Schule“ Stand heute 42 068 Dateien angesammelt. Darunter sind viele Unterrichtsdokumente: Arbeitsblätter, Bilder, Diagramme, Exkursionspläne, Fortbildungsskripte, Gefährdungsbeurteilungen, Hilfen und Lösungen, Klausuren und Tests, Lehrpläne, Präsentationen, Rückmeldebögen („Feedback“), Unterrichtsentwürfe, Versuchsvorschriften, Verweise auf Animationen, Podcasts, Videos und

Web-Adressen sowie anderes mehr. Um den Überblick zu behalten, was ich davon tatsächlich eingesetzt habe, und um das Material gegebenenfalls für den nächsten Gebrauch in folgenden Schuljahren kommentieren, ändern oder ersetzen zu können, fasse ich meine digitalen Dokumente in Unterrichtsskripten zusammen. In der Oberstufe umfasst solch ein Skript bei mir je eines der großen „Inhaltsfelder" wie Genetik, Neurobiologie usw., in der Sekundarstufe I den Unterricht einer Jahrgangsstufe, das ergibt dann 30 bis 50 Seiten. Anfangs habe ich die Skripte noch ohne Klausuren mit LibreOffice Writer erstellt.

Weil sich für mich ein einheitliches, gutes Schriftbild, Kopfzeilen, chemische Formeln und Diagramme leichter mit LaTeX-Mitteln erzielen lassen, bin ich dazu übergegangen, meine digitalen Unterrichtsdokumente einschließlich der Klausuren mit dem „LaTeX-Frontend" LyX[1] zusammenzufassen. Hierbei stieß ich auf folgendes Problem: Skripte der Oberstufe enthalten in der Regel vier mehrseitige Klausuren, je zwei pro Inhaltsfeld an den regulären Terminen und je zwei für die Nachschreibtermine. Eine Klausur umfasst, abhängig vom Fach und von der zeitlichen Nähe des Abiturs, zwei bis sechs Seiten (Aufgabenstellung und Material), die mit Seitenzahlen nach dem Muster „Seite x von y" versehen sind.

Auf diese Weise können sich die Schülerinnen und Schüler schnell von der Vollständigkeit der ausgehändigten Klausur überzeugen, auch wenn sie nicht von der Fachlehrperson ausgehändigt wird, und man kann sich bei der Besprechung auf die Seitenzahl beziehen. In den Unterrichtsskripten muss diese Seitennummerierung für jede Klausur lokal und unabhängig voneinander erfolgen. Gleichzeitig soll sich nicht die LaTeX-interne Seitenzählung als solche verändern, weil das Inhaltsverzeichnis noch zur Orientierung im Dokument zur Verfügung stehen soll.

Umsetzung mit LaTeX

Nach einer Weile des Recherchierens und Probierens merkte ich, dass meine LaTeX-Fertigkeiten zur Lösung des Problems nicht ausreichten. Daher wandte ich mich mit der Problemstellung und einem Minimalbeispiel an die deutschsprachige TeX-Nutzergruppe, und hier entstand durch Marei Peischl und Markus Kohm eine Lösung. Andere fanden die Umsetzung mehrerer lokaler Seitenzähler in einem Dokument auch elegant, und ich wurde gebeten, das Ergebnis hier vorzustellen.

Ausgehend von einem deutschsprachigen Dokument einer KOMA-Script-Klasse[2] – hier im Beispiel `scrbook` – wird das Paket `scrlayer-scrpage` geladen und die Felder der Kopf- und Fußzeilen werden geleert:

[1] `www.lyx.org`

[2] KOMA-Script ist meine persönliche Vorliebe. Ich habe es nicht ausprobiert, aber das hier erläuterte Vorgehen sollte sich auf die Standardklassen bei Verwendung von `fancyhdr` und auf die Memoir-Klasse übertragen lassen mit den entsprechenden Befehlen zur Gestaltung der Fußzeilen.

```
\documentclass{scrbook}
\usepackage[ngerman]{babel}
\usepackage{scrlayer-scrpage}
\clearpairofpagestyles
```

Nun werden zwei Zähler definiert. Der erste dient dazu, die Klausuren innerhalb des Dokuments zu zählen, der zweite zählt die Seiten innerhalb einer Klausur:

```
\newcounter{klausur}
\newcounter{klausurseite}
```

Nach der LaTeX-internen Zusammenstellung und Ausgabe einer Seite soll der zuvor definierte Zähler `klausurseite` um 1 erhöht werden. Dazu stellt das automatisch geladene Paket `lthooks` die Schnittstelle `\AddToHook` zur Verfügung, mit der der Befehl `\stepcounter` nachträglich angefügt werden kann:

```
\AddToHook{shipout/after}{\stepcounter{klausurseite}}
```

Für Klausuren wird eine Umgbung mit einer Variablen für den Titel der Klausur als Abschnittsüberschrift definiert. Der erste Teil dieser Umgebung beginnt mit dem Befehl `\clearpage` eine neue Seite nach Ausgabe vorhandener Gleitobjekte, erhöht den Klausurenzähler `klausur` um 1 und setzt den Zähler `klausurseite` auf 1. Mit dem Befehl `\cfoot` aus dem Paket `scrlayer-scrpage` wird die aktuelle und die letzte Klausurseite in den Seitenfuß geschrieben:

```
\newenvironment{klausur}[1]{% bei Beginn der Umgebung klausur
\clearpage
\stepcounter{klausur}
\setcounter{klausurseite}{1}
\cfoot{Seite \theklausurseite{} von \ref{klausur.\theklausur.letzteseite}}
\section{Klausur #1}
}
```

Dabei gibt `\theklausurseite` den Stand des Zählers `klausurseite` aus, und `\ref` greift auf eine Marke zu, die bei Beendigung der Umgebung erzeugt wird.

Im zweiten Teil der Umgebungsdefinition wird als Erstes wie zuvor eine neue Seite nach Ausgabe von Gleitobjekten begonnen. Der Zähler `klausurseite` ist jetzt um 1 größer als die Anzahl der Seiten der Klausur und muss daher um 2 vermindert werden. Warum um 2? Der anschließende Befehl `\refstepcounter` erhöht den Zähler wieder um 1, dieser hat damit den gesuchten Wert der letzten Klausurseite. Gleichzeitig macht `\refstepcounter` im Unterschied zu `\stepcounter` und `\addtocounter` diesen Wert für Marken und zur Referenzierung verfügbar; die Nummer der letzten Klausurseite kann wie in Code-Zeile 19 mit `\label` gespeichert werden.

```
{% am Ende der Umgebung klausur
\clearpage
\addtocounter{klausurseite}{-2}
\refstepcounter{klausurseite}
\label{klausur.\theklausur.letzteseite}
}
```

Dabei stellt der Zähler `klausur` sicher, dass jede Klausur eine eigene Marke erhält. Auf diese Marke greift `\ref` zu, um die Nummer der letzten Seite in die Fußzeile der Klausur zu schreiben. Nun kann jede Klausur mit eigenem Seitenzähler nach dem gewünschten Muster „Seite x von y“ als Abschnitt eines größeren Dokuments gestaltet werden, und die korrekte Seitenzählung für das gesamte Dokument bleibt im Inhaltsverzeichnis erhalten:

```
\begin{document}
…
\begin{klausur}{Grundbegriffe der Genetik}
…
\end{klausur}
…
\end{document}
```

Verwendung mehrerer lokaler Seitenzähler mit LyX

Ich bin ein LaTeX-Anwender, der gerne möglichst viel von dem, was er da zusammenstellt, bereits während der Erstellung an das erstrebte Ziel angenähert sehen möchte und der sich nach langen Schultagen am Abend unter Zeitdruck an Makros und längeren Code-Passagen im Quelltext des Dokuments deutlich weniger erfreuen kann als an deren Ergebnis. Wie für mich gemacht scheint mir deswegen LyX zu sein, eine Art „Frontend“ für LaTeX. Mit LyX kann ich mir in den Ferien mit Muße passenden LaTeX-Code suchen und ausprobieren, und in der Schulzeit kann ich den Code effektiv einsetzen, ohne dass er mich ablenkt.

Verwendung ohne weitere Anpassung

Zur Verwendung mehrerer lokaler Seitenzähler mit LyX öffne ich die Dokumenteinstellungen, wähle als Dokumentklasse eine der KOMA-Script-Klassen und kopiere die im vorangehenden Abschnitt erläuterten Code-Zeilen 3–20 in den „LaTeX-Vorspann“. Nun könnte ich bereits eine Klausur im Dokument mit einem lokalen Seitenzähler versehen, indem ich, wie in Abbildung 1 links gezeigt, mit Strg+L die Klausurumgebung mit dem LaTeX-Code `\begin{klausur}{Titel}` starte und mit

Abb. 1: Links ist die Verwendung eines lokalen Seitenzählers in LyX ohne weitere Anpassung gezeigt. Rechts ist die Verwendung eines lokalen Seitenzählers mit einer benutzerdefinierten Einfügung gezeigt, wie sie im Text erläutert wird.

`\end{klausur}` abschließe. Dieser Weg hat aber zwei Nachteile: Erstens muss man doch Code tippen, zweitens stimmt die Nummerierung der Abschnitte in LyX nicht mit der Nummerierung im kompilierten Dokument überein.

Verwendung mit einer benutzerdefinierten Einfügung

Wenn diese Nachteile eine Rolle spielen, kann man sie mit einer benutzerdefinierten Einfügung umgehen. Dazu ersetze ich in den Dokumenteinstellungen von LyX die Code-Zeilen im LaTeX-Vorspann durch die folgenden:

```
1 \usepackage{scrlayer-scrpage}
2 \clearpairofpagestyles
3 \newcounter{bereichnr}
```

```
\newcounter{seitenr}
\AddToHook{shipout/after}{\stepcounter{seitenr}}
\newcommand{\startsz}[1][Klausur]{
\clearpage
\setcounter{seitenr}{1}
\stepcounter{bereichnr}
\cfoot{#1 Seite \theseitenr{} von \ref{bereichnr.\thebereichnr.lastpage}}
}
\newcommand{\stoppsz}{
\clearpage
\addtocounter{seitenr}{-2}
\refstepcounter{seitenr}
\label{bereichnr.\thebereichnr.lastpage}
}
```

Es wird also statt einer Klausurumgebung sehr ähnlich ein Befehl `\startsz` definiert, um den Seitenzähler zu starten, und ein Befehl `\stoppsz`, um den Seitenzähler zu stoppen. Der Befehl `\startsz` hat eine optionale Variable zur Bezeichnung des nummerierten Seitenbereichs, die mit dem Wert Klausur vorbelegt ist. Auf diese Weise können zum Beispiel auch mehrseitige Versuchsanleitungen angemessen bezeichnet durchnummeriert werden.

Um auf die beiden so definierten Befehle in LyX ohne Code bequem zugreifen zu können, muss ich der Anwendung Namen (`LatexName startsz` bzw. `LatexName stoppsz`), Art (`LyxType custom` und `LatexType command`), Bezeichnung (`LabelString "Seitenzähler: Start"` bzw. `LabelString "Seitenzähler: Stopp"`), Darstellung (`Decoration Classic`) und die Option zum Ändern des Bezeichners (siehe Zeilen 9 bis 12 des folgenden Listings) mitteilen.

Dazu schreibe ich diese Angaben in den Dokumenteinstellungen von LyX in das Eingabefeld „Lokales Format“. Am Anfang steht die Deklaration des Formats der LyX-Datei, die ich dem LyX-Handbuch “LyX-Anpassung“ entnommen habe, dann werden die beiden Befehle in eine „flexible InsetLayout-Marke“ gehüllt. Die beiden InsetLayout-Marken erhalten ebenfalls selbsterklärende Bezeichner:

```
Format 66

InsetLayout "Flex:Seitenzähler: Start"
LyxType     custom
Decoration    Classic
LabelString   "Seitenzähler: Start"
LatexName     startsz
LatexType     command
Argument 1
```

```
LabelString   "Standardbezeichner ändern"
Mandatory     0
EndArgument
End

InsetLayout "Flex:Seitenzähler: Stopp"
LyxType     custom
Decoration    Classic
LabelString   "Seitenzähler: Stopp"
LatexName     stoppsz
LatexType     command
End
```

Dieses neue lokale Format muss über den Knopf *Validieren* von LyX geprüft werden und sollte „Format gültig!“ ergeben. Nach Bestätigung mit *OK* tauchen die Punkte *Seitenzähler: Start* und *Seitenzähler: Stopp* im Menü *Einfügungen* > *Benutzerdefinierte Einfügungen* auf und erscheinen nach Auswahl in LyX wie in Abbildung 1 rechts gezeigt. Die kompilierten Ergebnisse sind bei direkter Verwendung von LaTeX sowie mit und ohne benutzerdefinierte Einfügung in LyX gleich. Allerdings darf in LyX aus mir nicht bekannten Gründen der Seitenzähler nicht ganz am Ende des Dokuments stoppen, andernfalls wird „Seite x von ??“ ausgegeben.

Praktisch sind Tastaturkürzel zum Starten und Stoppen eines Seitenzählers. Diese werden angelegt, indem in den LyX-Einstellungen (im Menü *Werkzeuge*) in der Rubrik Tastenkürzel der Knopf *Neu* gewählt wird. In der sich öffnenden Maske gibt man als Funktion `flex-insert "Seitenzähler: Start"` bzw. `flex-insert "Seitenzähler: Stopp"` ein, ordnet ein Tastenkürzel zu und bestätigt mit *OK*.

VSCodium – Eine Entwicklungsumgebung

Rainer-M. Fritsch

VSCodium stellt eine Entwicklungsumgebung zur Verfügung, die man fast nie verlassen muss. Dateimanagement, integrierter PDF-Viewer, Anzeige der Warnungen und Fehler eines LaTeX-Laufs im Quelltext, vielfältig anpassbar und erweiterbar. Und das über alle Betriebssysteme und Endgeräte hinweg synchronisierbar – ohne Medienbrüche kann nahtlos an Projekten weitergearbeitet werden. Wer neben oder in LaTeX-Projekten mit HTML/CSS, JavaScript, Asciidoc, Python oder Ähnlichem arbeitet, wird viel Nutzen aus VSCodium als Entwicklungsumgebung ziehen können.

Die Frage, welcher »der richtige Editor« für LaTeX sei, ist schwer zu beantworten. Die einen schwören auf Emacs oder Vim oder auf spezielle LaTeX-Editoren wie Kile, TeXmaker, TeXshop usw. Die letzteren sind Editoren, die spezielle Aufgaben erfüllen und dafür sicherlich auch optimiert sind. Universelle Editoren, die mehrere Computer-Sprachen beherrschen, halte ich jedoch für günstiger, als sich in verschiedene Editoren für verschiedene Aufgaben und Sprachen einzuarbeiten.

Es lohnt, sich mit VSCodium zu beschäftigen – gerade dann, wenn mann/frau viel schreiben, sich in verschiedenen Sprachen und unterschiedlichen Betriebssystemen bewegen muss und seine/ihre Projekte in einer einzigen Entwicklungsumgebung organisieren will.

VSCodium basiert auf der Open-Source-Software Visual Studio Code von Microsoft – meist im Netz als VS Code verkürzt.[5]
»Microsofts vscode-Quellcode ist quelloffen (MIT-lizenziert), aber das zum Download verfügbare Produkt (Visual Studio Code) ist unter dieser nicht-[Free/Libre Open Source Software]-Lizenz lizenziert und enthält Telemetrie/Tracking.«[1] [8]

VSCodium ist ein moderner Editor, der alle Funktionen auf sich vereinigt, die in der täglichen Arbeit mit Auszeichnungs- und Programmiersprachen erwartet werden. Der Editor beherrscht Syntaxhervorhebung, Multi-Cursor, Code-Faltung, Debugging, automatische Vervollständigung bei der Bearbeitung von Code sowie Text (IntelliSense) und die Versionsverwaltung mit `git` ist bereits vorkonfiguriert. VSCodium unterstützt mehr als 30 unterschiedliche Programmier-, Auszeichnungs- und Datenbanksprachen. Für den Einstieg gibt es ein kostenloses eBook von Microsoft-Mitarbeitern [3] und ein kostenloses Video – ebenfalls auf Deutsch – bei Udemy [7] (Registrierung erforderlich).[2] Hilfreich für einen schnellen Einstieg ist auch »Visual Studio Code Crash Course« (englisch).[6]

Was macht VSCodium so einzigartig? VSCodium stellt eine Entwicklungsumgebung zur Verfügung, die man fast nie verlassen muss. Dateimanagement, integrierter PDF-Viewer, Anzeige der Warnungen und Fehler eines LaTeX-Laufs im Quelltext, vielfältig anpassbar und erweiterbar – auch dank der vielen ausgereiften Erweiterungen und Designs der Benutzeroberfläche. Und das über alle Betriebssysteme und Endgeräte hinweg synchronisierbar. Egal ob unter Linux, Windows oder MacOS oder ob auf dem Notebook, dem stationären PC oder seit neuestem per Webbrowser sogar auf dem Tablet[3]; ohne Medienbrüche kann nahtlos an den eigenen Projekten weitergearbeitet werden.[4] Wer neben oder in LaTeX-Projekten mit HTML/CSS,

[1] Visual Studio Code sammelt Telemetriedaten wie Absturzberichte, Fehlertelemetrie sowie Nutzungsdaten wie Informationen darüber, wie Funktionen in VS Code verwendet werden und funktionieren.
[2] Das Video ist in vielen Teilen auch bei YouTube `https://youtu.be/K2P5oyCIZek` verfügbar.
[3] `https://vscode.dev/`
[4] Unter Windows und Linux sind viele Tastenkombinationen gleich, unter MacOS wird statt der ctrl -Taste oft die cmd -Taste benutzt.

JavaScript, AsciiDoc, Python o. ä. arbeitet, wird viel Nutzen aus VSCodium als gemeinsamer Entwicklungsumgebung ziehen können.

Wie VSCodium für LaTeX eingesetzt und angepasst werden kann, soll im Folgenden gezeigt werden.

VSCodium (Visual Studio Code) für LaTeX einrichten

VSCodium einrichten

VSCodium steht für Linux, Windows und MacOS zur Verfügung und stellt alle Eigenschaften und Werkzeuge bereit wie Visual Studio Code von Microsoft. Beim ersten Start der Entwicklungsumgebung kann die deutschsprachige Benutzeroberfläche geladen und installiert werden.[5]

Des Weiteren wird nach dem gewünschten Farbdesign (Theme) gefragt. Hier kann zwischen verschiedenen hellen und dunklen Farbdesigns gewählt werden. Weitere Themes können nachinstalliert werden.

Das Hauptfenster teilt sich grob in drei Bereiche (siehe Abb. 1, S. 41): Links die `Activity Bar` und der `Explorer`, in der Mitte der Bereich des `Editors` mit dem jeweiligen Quelltext und rechts der `PDF-Viewer`. Statt des PDF-Viewers kann auch ein weiteres Editor-Panel eingeblendet werden, wenn man z. B. zwei Dateien mit einander vergleichen möchte. Oberhalb der `Status Bar`/unterhalb des Editierbereichs kann ein weiteres Panel z. B. für ein Terminal-Fenster eingeblendet werden.

Es gibt über 27 000 Erweiterungen. Alle Erweiterungen von VS Code können auch in VSCodium eingesetzt werden. Diese werden jedoch nicht wie bei VS Code über den Marktplatz von Microsoft[6] geladen, sondern über die Open VSX Registry.[7] Dazu später mehr im zweiten Teil dieses Beitrags.

Fürs erste sollten folgende Erweiterungen installiert werden:

- Code Spell Checker
- German – Code Spell Checker
- vscode-icons (für schönere Icons im Datei-Explorer)

Dazu klickt man in der `Activity Bar` links auf das Symbol mit den vier Quadraten und gibt in der Suche nach Erweiterungen die o. g. Erweiterungen ein.

Soweit `git` als Versionskontrolle noch nicht installiert ist, sollte dies jetzt nachgeholt werden. Mehr zu `git` und wie es installiert wird, findet sich unter [2].

[5] German Language Pack for Visual Studio Code
[6] `https://marketplace.visualstudio.com/VSCode`
[7] `https://open-vsx.org/`

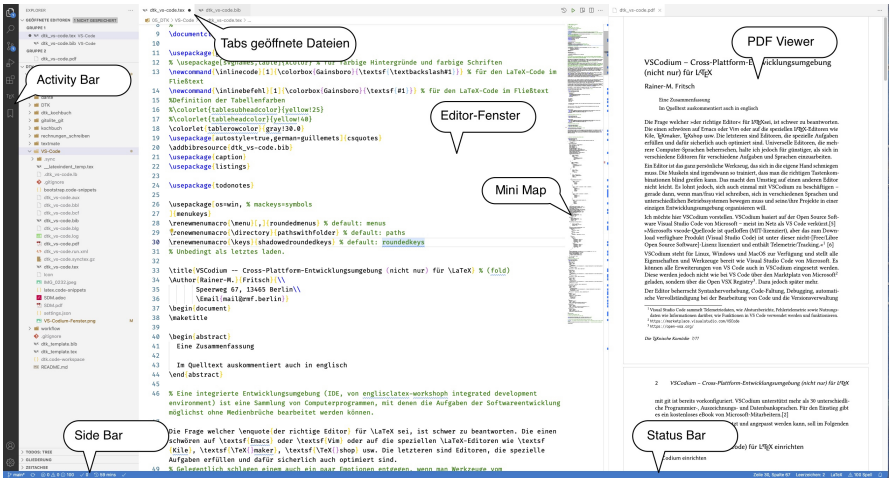


Abb. 1: Hauptfenster VSCodium

Erweiterungen: LaTeX Workshop und BibLaTeX Snippets

`LaTeX Workshop` von James Yu ist ein unglaublich vielseitiges Werkzeug für die Arbeit mit LaTeX.[9] Zusammen mit den vielen Möglichkeiten von VSCodium als Entwicklungsumgebung ist ein konzentriertes und effizientes Arbeiten möglich. Diese Erweiterung ist ausführlich in einem GitHub-Wiki dokumentiert.[10] Wie die Arbeitsumgebung den persönlichen Bedürfnissen angepasst werden kann, wird im nächsten Abschnitt gezeigt.

Ergänzt wird `LaTeX Workshop` mit der Erweiterung `BibLaTeX Snippets`. Mit der Tastenkombination ctrl + Leertaste werden in einer *.bib-Datei die verschiedensten vordefinierten Quellen von `article` bis `xdata` angeboten und mit den entsprechend vordefinierten Angaben in die *.bib-Datei eingetragen. Mit →| kann von Wert zu Wert gesprungen werden, um die entsprechenden Quellenangaben (key, author, title usw.) einzutragen.

Es gibt weitere sehr hilfreiche und empfehlenswerte Erweiterungen für Projekte mit LaTeX. Diese werden im zweiten Teil dieses Beitrags vorgestellt.

Nach der Installation dieser beiden Erweiterungen erscheint an sechster Stelle der `Activity Bar` ein neues Icon TEX. Mit einem Klick auf dieses Icon werden im `Explorer` COMMANDS, STRUCTURE und SNIPPET VIEW mit weiteren Untermenüs

angeboten. Es ist eine Frage des persönlichen Geschmacks, ob man diese Befehle oder eher schnelle Tastenkombinationen nutzen möchte.[8]

Ein LaTeX-Projekt erstellen

Datei- und Projektmanagement mit VSCodium

VSCodium verwaltet Projekte in sog. »worksspaces« (Arbeitsbereiche). Jedem Arbeitsbereich können Verzeichnisse und Dateien sowie individuelle Einstellungen zugeordneten werden, die nur für diesen Arbeitsbereich gelten sollen.

Jedem Arbeitsbereich muss mindestens ein Verzeichnis hinzugefügt werden, dass die Projektdateien beinhaltet. Die Verschachtelungstiefe und das Hinzufügen von weiteren Verzeichnissen scheint unbegrenzt. Nicht alle Verzeichnisse müssen unterhalb des Projektverzeichnisses liegen. So können Verzeichnisse auch aus anderen Projekten hinzugefügt werden, wenn auf gleiche *.bib-Dateien oder Bild-Verzeichnisse usw. zugegriffen werden soll (Menü Datei » Ordner zu Arbeitsbereich hinzufügen).

Der Arbeitsbereich wird unter Menü Datei » Arbeitsbereich speichern unter als `IhrProjekt.code-workspace` im Projektverzeichnis gespeichert.

Bei einem Klick auf das dritte Symbol von oben (`Quellcodeverwaltung`) in der `Activity Bar` kann für das neue Projekt ein git-Repository initialisiert und dieses Projekt ggf. mit einem GitHub-Konto verbunden werden (siehe Abb. 1, S. 41). Sollten Verzeichnisse unterschiedlichen git-Repositories angehören, werden diese Repositories innerhalb von VSCodium auch differenziert verwaltet.

Die Symbole in der `Activity Bar` haben zusätzlich die Funktion, das `Explorer`-Fenster auf- und zuzufalten (Tastenkürzel ctrl + b). Ein Klick auf das oberste Symbol in der `Activity Bar` zeigt im `Explorer` den Arbeitsbereich des Projektes mit allen Dateien und Verzeichnissen. Ein Klick auf eine Datei öffnet diese. Dabei gibt es folgende Besonderheit zu beachten: Bei einem Klick auf eine andere Datei wird die zuletzt geöffnete Datei geschlossen – wenn diese nicht editiert wurde – und die neue Datei geöffnet. Ein Doppelklick auf eine Datei öffnet diese dauerhaft, bis diese von Hand geschlossen wird.

Im Explorer stehen alle Funktionen für das Dateimanagement zur Verfügung. Berührt man mit der Maus im `Explorer` den horizontalen Balken mit dem Namen des Arbeitsbereichs, werden Symbole für das Anlegen neuer Dateien und Verzeichnisse sichtbar. In großen Projekten mit vielen Unterverzeichnissen ist das Symbol rechts zum Zufalten aller geöffneten Verzeichnisse sehr hilfreich. Ein Rechtsklick auf ein Verzeichnis oder eine Datei stellte weitere Funktion zum Dateimanagement

[8] Eine Übersicht der Tastenkombinationen/shortcuts von VSCodium/VS Code unter `https://code.visualstudio.com/shortcuts/keyboard-shortcuts-linux.pdf` oder ctrl + s + ctrl + k

zur Verfügung. Ein Rechtsklick auf eine Datei öffnet auch eine `Zeitachse` für die `git`-Historie einer Datei.

Alle Einstellungen für den Editor wie Schriftgröße, die Schriftart, das Verhalten des Editors oder für die Erweiterungen können unter den Einstellungen (ctrl + ,) oder mit Klick auf das Zahnrad-Symbol in der `Activity Bar` getroffen werden. Es öffnet sich ein weiterer Tab, in dem sehr übersichtlich alle Einstellungen mit Erläuterungen gezeigt und angepasst werden können. Wichtig ist der Hinweis, dass die Einstellungen benutzerspezifisch, auf den Arbeitsbereich oder nur auf das Verzeichnis bezogen gespeichert werden können (man beachte die drei Tabs unter dem Suchfeld für die Einstellungen).

Die Einstellungen werden in einer Datei `settings.json` gespeichert. Diese kann auch direkt bearbeitet werden. Man beachte dazu im Einstellungs-Tab das dritte Symbol von links in der Tab-Zeile am rechten Rand. Mit einem Klick auf dieses Symbol sieht man auch den Pfad zur Datei `settings.json`.

Die Pfade für die `settings.json` für den Benutzer sind je nach Betriebssystem hier zu finden

– für VS Code:

- Linux: $HOME/.config/Code/User/settings.json
- macOS: $HOME/Library/Application Support/Code/User/settings.json
- Windows: %APPDATA%\Code\User\settings.json

– für VSCodium:

- Linux: $HOME/.config/VSCodium/User/settings.json
- macOS: $HOME/Library/Application Support/VSCodium/User/settings.json
- Windows: %APPDATA%\VSCodium\User\settings.json

Die auf den Arbeitsbereich bezogenen Einstellungen werden im Wurzelverzeichnis des Arbeitsbereichs in einem verstecktem Verzeichnis `.vscode` abgelegt.[9]

Arbeiten mit dem Editor und IntellSense

Die Erweiterung `LaTeX Workshop` stellt für das Hilfsmittel `IntelliSense` alles bereit, um bei der Eingabe in LaTeX-Dokumente viel Schreibarbeit zu vermeiden. Alle LaTeX-Befehle stehen zu Verfügung; Klammern werden automatisch gesetzt und der Cursor springt zwischen die Klammern, um weitere Werte entgegenzunehmen. Das Eingeben weniger Zeichen genügt, dass ein Auswahlmenü rechts vom Cursor

[9] In VS Code können die Einstellungen, die Tastenkombinationen, Benutzercodeausschnitte, Erweiterungen und der Benutzeroberflächenzustand über ein Microsoft- oder GitHub-Konto synchronisiert werden. Diese Funktion ist in VS Code unter dem Zahnrad-Symbol in der Activity Bar zu finden. Für VSCodium hilft die Erweiterung `Settings Sync` weiter.

erscheint. Mit den Pfeiltasten kann ausgewählt und mit [→|] oder [↵] bestätigt werden. Das gilt auch für beliebige Wörter, die bereits im Text geschrieben wurden. Fachtermini müssen nicht immer wieder neu ausgeschrieben werden, sondern kommen per [→|] nach wenigen Eingaben der ersten Buchstaben in den Text. Je häufiger Wörter benutzt werden, desto höher stehen diese im Auswahlmenü.

Bei Umgebungen kann gleich die Art der Umgebung ausgewählt werden und das Ende der Umgebung wird automatisch gesetzt. Der Cursor springt in eine Leerzeile zwischen `\begin` und `\end` und die Eingabe kann ungestört fortgeführt werden.

Bei Umgebungen für Aufzählungen wird nach jeder neuen Zeile automatisch ein `\item` je nach Art der Aufzählung eingefügt, einschließlich eckiger Klammern z. B. bei der `description`-Umgebung. Korrekte Einrückungen für eine bessere Lesbarkeit des Quellcodes sind selbstverständlich. Man gewöhnt sich unglaublich schnell an diese unterstützenden Funktionen, dass diese bald in jedem anderen Editor oder Textverarbeitungsprogramm vermisst werden.

Die Minimap rechts vom Editor (siehe Abb. 1, S. 41) zeigt nicht nur an, in welchem Bereich des Quellcodes man sich derzeit befindet, sondern nach dem ersten LaTeX-Lauf werden gelb Warnungen (z. B. overfull box) oder rot Fehler angezeigt. Hält man den Mauszeiger über eine der gelb oder rot unterstrichenen Zeilen, wird der Inhalt der entsprechenden Warnung oder des Fehlers ausgegeben.

Zu den Stärken des Editors gehört auch bei Referenzen `\ref` oder Zitaten `\cite` die bereits hinterlegten label oder keywords aus der *.bib-Datei als Kontextmenü anzuzeigen.

Unbekannte Wörter werden mit einer blauen Linie unterlegt, diese können per rechtem Mausklick dem Wörterbuch (des Benutzers, des Arbeitsbereichs oder des Verzeichnises) hinzugefügt oder mit [ctrl]+[.] entsprechend der dortigen Vorschläge korrigiert werden.

Den Quellcode übersetzen

Mit dem TeX-Symbol in der Activity Bar stehen bereits verschiedene Rezepte (Recipes) für das Übersetzen des LaTeX-Codes zur Verfügung. LaTeX-Workshop kann auch TeX-Magic comments verarbeiten. Mit

```
% !TEX TS-program = lualatex
% !BIB program = biber
```

am Anfang des Quellcodes werden automatisch vier Übersetzungsläufe gestartet, um alle Referenzen aufzulösen. Die Übersetzung wird mit der Tastenkombination [ctrl]+[alt]+[b] oder dem grünen Dreieck rechts oben in der Tab-Leiste ausgelöst.

Bei einem Fehler öffnet sich rechts unten ein kleines Fenster, das auf einen Fehler hinweist und anbietet, den Compiler-Log zu öffnen. Mit Klick auf dieses Fenster oder mit ctrl + ⇧ + m werden die Probleme mit Zeilennummern und Fehlertext angezeigt. Ein Klick auf den Fehler springt an die entsprechende Stelle im Quellcode.

In umfangreicheren Projekten mit Bibliothek, Glossar, Sachregistern usw. sind umfangreichere Übersetzungsläufe nötig. Diese können als Rezepte in den Einstellungen der Erweiterung LaTeX-Workshop hinterlegt werden. Vier Beispiele zeigt das folgende Listing, das in die `settings.json` eingefügt werden muss:

```
"latex-workshop.latex.recipes": [
      {
        "name": "xelatex*3",
        "tools": [
            "xelatex",
            "xelatex",
            "xelatex"
        ]
      },
      {
        "name": "xelatex -> glossaries -> biber -> xelatex*2",
        "tools": [
            "xelatex",
            "makeglossaries",
            "biber",
            "xelatex",
            "xelatex"
        ]
      },
      {
        "name": "xelatex -> glossaries -> xelatex*2",
        "tools": [
            "xelatex",
            "makeglossaries",
            "xelatex",
            "xelatex"
        ]
      },
      {
        "name": "lualatex->biber->lualatex x2",
        "tools": [
          "lualatex",
          "biber",
          "lualatex",
```

```
35            "lualatex"
36          ]
37      }
38    ],
```

Listing 1: Beispiele für Übersetzungsrezepte

Der Aufruf erfolgt über die Befehlspalette (ctrl + ⇧ + p); dort in das Suchfeld »Latex Workshop: Build with recipe« oder kurz »build« eingeben und eines der o. g. Rezepte auswählen. Oder man klickt in der Activity Bar auf das Icon TEX, dann COMMANDS › Build LaTeX project und wählt das entsprechende Rezept (Recipe) aus.

Nach der Übersetzung kann das PDF mit ctrl + alt + v angezeigt werden (siehe Abb. 1, S. 41). Das PDF wird nach jeder Übersetzung aktualisiert. Aus dem PDF springt man mit ctrl + ⇧ und Mausklick links an die entsprechende Stelle im Quellcode; mit ctrl + alt + j vom Quellcode an die entsprechende Stelle im PDF.

Das PDF kann auch über das Icon TEX in der Activity Bar, dann COMMANDS › View LaTeX PDF innerhalb von VSCodium, in einem Web Browser oder in einem externen PDF Viewer angezeigt werden.

Die Struktur der eigenen Arbeit wird nicht nur im PDF angezeigt, sondern innerhalb von VSCodium kann die Gliederung des LaTeX-Quelltextes über die Activity Bar Icon TEX › STRUCTURE angezeigt und per Mausklick zu den einzelnen Abschnitten im Quelltext navigiert werden.

Code-Snippets erstellen

Snippets stellen eine erhebliche Arbeitserleichterung dar. Die ausführliche Dokumentation findet sich unter [4]. Am Beispiel eines Abschnitts in LaTeX wird kurz die Funktionsweise von Snippets erläutert. Das Snippet soll den Namen eines Abschnitts in das Label übernehmen und als Kommentar das Ende eines Abschnitts anzeigen wie im folgenden Listing:

```
\subsection{Code-Snippets erstellen}
  \label{sub:Code-SnippetsErstellen}
% subsection Code-Snippets erstellen (end)
```

Soweit eine Snippet-Datei noch nicht existiert, wird diese über Menü Datei › Einstellungen › Benutzerausschnitte mit anschließender Auswahl der Sprache LaTeX erstellt. Die Snippets werden in einer latex.json-Datei gespeichert. Die Syntax der Snippets folgt im Wesentlichen der *TextMate snippet syntax*.[10]

[10] https://macromates.com/manual/en/snippets

Da VSCodium Multi-Cursor über mehrere Zeilen beherrscht, kann mit der Eingabe des Abschnitttitels gleichzeitig das Label und der Kommentar am Ende des Abschnitts editiert werden. Dabei soll im Label der Text in Kleinbuchstaben umgewandelt werden (s. Listing 2, Zeile 5).

```
1 "subsection-label":{
2 "prefix": "ssect",
3 "body": [
4   "\\subsection{$1}",
5 "\t\\label{sub:${1/(.*)/${1:/downcase}/}}",
6 // downcase funktioniert erst nach dem Tab gedrückt wurde.
7   "% subsection $1 (end)"
8 ],
9 "description": "subsection, label, subsection end"
```

Listing 2: Ein Code-Snippet für eine subsection erstellen

`\t` fügt einen Tab-Schritt ein. Aufgerufen wird das Snippet mit der Eingabe von »ssect« (`prefix` im Listing) oder mit dem Tastenkürzel ctrl + Leertaste. Auf diese Weise lassen sich weitere Snippets für Chapter, Section usw. erstellen.

Ein Beispiel für eine itemize-Umgebung:

```
1 "Itemize":{
2   "prefix": "Itemize",
3   "description": "Itemize Umgebung",
4   "body": [
5     "\\begin{itemize}",
6     "\t\\item ${1}",
7     "\\end{itemize}"
8   ]
9 },
```

Listing 3: Itemize-Umgebung mit wenig Aufwand erstellen

Meine vollständigen Dateien `settings.json` und `latex.json` mit meinen Snippets können unter [1] bei GitHub (auch mit einem Beispiel für Bootstrap-Snippets) heruntergeladen werden.

Die jeweiligen sprachspezifischen Snippets werden hier abgelegt

– für VS Code:

- Linux: $HOME/.config/Code/User/snippets/
- macOS: $HOME/Library/Application Support/Code/User/snippets/
- Windows: %APPDATA%\Code\User\snippets\

– für VSCodium:

- Linux: $HOME/.config/VSCodium/User/snippets/
- macOS: $HOME/Library/Application Support/VSCodium/User/snippets/
- Windows: %APPDATA%\VSCodium\User\snippets\

Versionsverwaltung mit `git`

Das dritte Symbol von oben in der `Activity Bar` ist die Quellcodeverwaltung (ctrl + ⇧ + g). Das Symbol bekommt einen »Störer« mit der Anzahl der geänderten Dateien. Wenn man in der Quellcodeverwaltung auf eine geänderte Datei klickt, werden die Änderungen der Datei im Vergleich zur Vorversion in zwei nebeneinanderliegenden Editor-Fenstern angezeigt – links die letzte, rechts die aktuelle Version. Dort sind Löschungen rot und Einfügungen grün markiert.

Im Eingabefeld »Nachricht« im `Explorer` über den geänderten Dateien schreibt man einen Kommentar für den `commit`. Mit Klick auf das Symbol √ wird die Datei in das Repository geschrieben.

Der »Störer« auf dem Symbol der Quellcodeverwaltung erinnert beständig daran, dass die geänderte Datei noch nicht im Repository gesichert wurde. In VSCodium ist eine einfache und intuitive Versionsverwaltung im ersten Schritt realisiert. Weitere Informationen zu `git` und seinen vielfältigen Möglichkeiten sind unter [2] zu finden.

Weitere nützliche Erweiterungen

Weitere nützliche Erweiterungen mit Hinweisen für die Installation und Konfiguration werden in der nächsten Folge dieses Beitrages vorgestellt. Dazu gehören u. a.:

- Advance New File
- Bookmarks
- Bracket Pair Colorizer 2
- Git History
- Insert Date String
- Live Share
- TODO Tree

Literatur

[1] Rainer-Maria Fritsch: Beispiel-Dateien für DTK, 2022, `https://github.com/rmfberlin/TeXLaTeX/tree/main/VSCodium` (besucht am 2. 12. 2021).

[2] git-scm.com: Git – fast-version-control, 2021, `https://git-scm.com/` (besucht am 30. 11. 2021).

[3] Kay Giza, Tobias Kahlert: eBook: Visual Studio Code Tipps & Tricks Vol. 1, 2016, http://aka.ms/ebook-VSCodeTippsTricks (besucht am 27. 11. 2021).

[4] Microsoft: Snippets in Visual Studio Code, 2021, https://code.visualstudio.com/docs/editor/userdefinedsnippets (besucht am 3. 12. 2021).

[5] — Visual Studio Code – Code Editing. Redefined. 2021, https://code.visualstudio.com/ (besucht am 26. 11. 2021).

[6] James Q Quick: Visual Studio Code Crash Course, 2020, https://www.youtube.com/watch?v=WPqXP_kLzpo.

[7] Jan Schaffranek: Einführung in Visual Studio Code, 2021, https://www.udemy.com/course/einfuhrung-in-visual-studio-code/ (besucht am 30. 11. 2021).

[8] vscodium community: VSCodium – Free/Libre Open Source Software Binaries of VS Code, 2021, https://vscodium.com/ (besucht am 26. 11. 2021).

[9] James Yu: LaTeX Workshop, 2021, https://open-vsx.org/extension/James-Yu/latex-workshop (besucht am 18. 4. 2020).

[10] — LaTeX-Workshop wiki, 2021, https://github.com/James-Yu/LaTeX-Workshop/wiki (besucht am 30. 11. 2021).

Schriften für XeLaTeX und LuaLaTeX

Herbert Voß

Für beide, nun nicht mehr allzu neuen TeX-Programme gibt es im Zusammenhang mit Schriften einige Dinge zu beachten, die für Anwender wichtig sind, die bislang mit pdfLaTeX gearbeitet haben. Denn bisher galt, dass für »unbedarfte« Anwender nur die Schriften zur Verfügung standen, die die jeweilige TeX-Distribution mitgeliefert hat.

Mit der Anwendung von XeLaTeX oder LuaLaTeX sind folgende Dinge zu beachten:

- Die Standardschrift ist per Definition Latin Modern, unabhängig davon, ob `fontspec` geladen wird. Für pdfLaTeX war dies bislang die Computer Modern.
- Die Pakete `inputenc` und/oder `fontenc` sollten nicht geladen werden. Zum einen ist UTF-8 die Standardeingabekodierung und zum anderen lädt `fontspec` automatisch `fontenc` mit der Kodierung `TU` (Unicode).
- Für die Unterstützung von mathematischen OpenType-Schriften ist das Paket `unicode-math` anstelle der Pakete `amsfonts` und/oder `amssymb` zu laden. Das Paket `amsmath` kann nach wie vor benutzt werden, sollte jedoch *vor* `fontspec`/`unicode-math` geladen werden.

- Die Pakete `xltxtra` und `xunicode` sind veraltet und sollten nicht mehr verwendet werden.

Sowohl mit XeLaTeX als auch mit LuaLaTeX kann das Paket `fontspec` verwendet werden: `\usepackage[Optionen]{fontspec}`. Es erleichtert wesentlich das Einbinden von OpenType- und TrueType-Schriften, die nicht Teil der TeX-Distribution sind. Damit sie auch vom System *automatisch* gefunden werden, müssen diese entweder im aktuellen Dokumentenverzeichnis oder in Abhängigkeit vom Betriebssystem in folgenden Verzeichnissen liegen:

Linux	`/usr/share/fonts`, `/usr/local/share/fonts`, `~/.fonts` (nur User)
Windows	`C:\Windows\Fonts`
Mac OS X	`/System/Library/Fonts`, `/Library/Fonts`, `~/Library/Fonts` (nur User)

Die Eigenschaften einer Schrift kann man sich mit einem entsprechenden Programm, beispielsweise `otfinfo` ausgeben lassen:

```
bash-3.2$ otfinfo -i `kpsewhich ComicNeue-Regular.otf`
Family:              Comic Neue
Subfamily:           Regular
Full name:           Comic Neue Regular
PostScript name:     ComicNeue-Regular
Version:             Version 2.003;hotconv 1.0.109;makeotfexe 2.5.65596
Unique ID:           2.003;;Comic Neue Light
Designer:            Craig Rozynski
Designer URL:        http://www.craigrozynski.com
Manufacturer:        Craig Rozynski
Vendor URL:          http://www.comicneue.com
Copyright:           Copyright 2014 The Comic Neue Project Authors (https://github.com/
 ↪crozynski/comicneue)
License URL:         https://scripts.sil.org/OFL
License Description: This Font Software is licensed under the SIL Open Font License,
 ↪Version 1.1. This license is available with a FAQ at: https://scripts.sil.org/OFL
Vendor ID:           UKWN
```

Schriftsuche

Mit dem Programm `luaotfload-tool` kann man auf dem System installierte Schriften auflisten. Dabei werden sowohl die Systemschriften als auch die TeX-Schriften beachtet. Allerdings ist das Programm mehr auf die Bedürfnisse von TeX selbst als auf die der Anwender gerichtet. Beispielsweise liefert die Suche nach der Schrift Times:

```
bash-3.2$ luaotfload-tool --find=times
luaotfload | resolve : Font "times" found!
luaotfload | resolve : Resolved file name "/System/Library/Fonts/Times.ttc",
 ↪subfont nr. 0
```

Die Ausgabe liefert nur eine Schrift, obwohl mehrere Varianten installiert sind, allerdings unter verschiedenen Dateinamen. Die Suche lässt sich verfeinern, indem man die Option `--fuzzy` verwendet und jetzt beispielsweise nach »times new roman« sucht. Aber auch diese Suche ist nicht sonderlich erfolgreich. Mit dem vorhandenen Lua-Skript `luafindfont`, welches Teil der TeX-Distribution ist, und der Suche nach *times* erhält man dagegen ein besseres Ergebniss:

```
bash-3.2$ luafindfont times
We are using Lua 5.3
Looking for font "times"
Check for file /usr/local/texlive/2021/texmf-var/luatex-cache/generic/names/luaotfload-names
 No.                        Fontname      Symbolic Name                                Path
   1.  Times New Roman Bold Italic.ttf      timesnewroman     /Users/voss/Library/Fonts/Times/
   2.  Times New Roman Bold Italic.ttf      timesnewroman  /System/Library/Fonts/Supplemental/
   3.         Times New Roman Bold.ttf      timesnewroman  /System/Library/Fonts/Supplemental/
   4.         Times New Roman Bold.ttf      timesnewroman     /Users/voss/Library/Fonts/Times/
   5.       Times New Roman Italic.ttf      timesnewroman     /Users/voss/Library/Fonts/Times/
   6.       Times New Roman Italic.ttf      timesnewroman  /System/Library/Fonts/Supplemental/
   7.              Times New Roman.ttf      timesnewroman  /System/Library/Fonts/Supplemental/
   8.              Times New Roman.ttf      timesnewroman     /Users/voss/Library/Fonts/Times/
   9.                        Times.ttc              times              /System/Library/Fonts/
  10.            Times_Sans_Serif.ttf     timessansserif     /Users/voss/Library/Fonts/Times/
  11.      TimesNewRomanMTStd-Bold.otf timesnewromanmtstd     /Users/voss/Library/Fonts/Times/
  12.  TimesNewRomanMTStd-BoldCond.otf timesnewromanmtstd     /Users/voss/Library/Fonts/Times/
  13.    TimesNewRomanMTStd-BoldIt.otf timesnewromanmtstd     /Users/voss/Library/Fonts/Times/
  14.      TimesNewRomanMTStd-Cond.otf timesnewromanmtstd     /Users/voss/Library/Fonts/Times/
  15.    TimesNewRomanMTStd-CondIt.otf timesnewromanmtstd     /Users/voss/Library/Fonts/Times/
  16.    TimesNewRomanMTStd-Italic.otf timesnewromanmtstd     /Users/voss/Library/Fonts/Times/
  17.           TimesNewRomanMTStd.otf timesnewromanmtstd     /Users/voss/Library/Fonts/Times/
  18. TimesNewRomanPS-BoldItalicMT.otf  timesnewromanpsmt     /Users/voss/Library/Fonts/Times/
  19.       TimesNewRomanPS-BoldMT.otf  timesnewromanpsmt     /Users/voss/Library/Fonts/Times/
  20.            TimesNewRomanPSMT.otf  timesnewromanpsmt     /Users/voss/Library/Fonts/Times/
  21.      TimesNewRomanPSStd-Bold.otf timesnewromanpsstd     /Users/voss/Library/Fonts/Times/
  22.    TimesNewRomanPSStd-BoldIt.otf timesnewromanpsstd     /Users/voss/Library/Fonts/Times/
  23.    TimesNewRomanPSStd-Italic.otf timesnewromanpsstd     /Users/voss/Library/Fonts/Times/
  24.   TimesNewRomanPSStd-Regular.otf timesnewromanpsstd     /Users/voss/Library/Fonts/Times/
```

Das Programm kann mit verschiedenen Parametern gestartet werden und erlaubt bei der Angabe der Schrift auch eine Und-Bedingung.

Parameter

```
-h,--help
-o,--otfinfo (default 0)
-i,--info (default 0)
-n,--nosymbolicnames
-v...          Verbosity output
-m,--max_string (default 90)
<font> (string)
```

`-m <Anzahl>` Anzahl der Zeichen, die für die Ausgabe des Schriftnamens *einschließlich* Pfad berücksichtigt werden. Die Angabe kann sehr lang sein und kann durch Angabe von beispielsweise `-m 50` auf 50 Zeichen beschränkt werden. Dabei werden Zeichen in der Mitte der Zeichenkette unterdrückt und durch … ersetzt.

-o ‹Nummer[Option]› Für die Schrift mit der angegebenen Nummer wird das Programm `otfinfo` angewendet und liefert dann ergänzende Schriftinformationen. Optionen für `otfinfo` müssen direkt der Schriftnummer folgen.
-i ‹Nummer› Für die Schrift mit der angegebenen Nummer werden die vorhanden Schriftschnitte ausgegeben.
-n Entfallen der mittleren Spalte bei der Ausgabe.

Mögliche Anwendungen wären beispielsweise:

```
luafindfont times               # Suche nach times im Namen
luafindfont palatino -o 3       # Suche nach palatino und starte otfinfo mit Schrift Nr.3
luafindfont -i 3 -m 50 arial    # Suche nach arial, Schriftschnitte Nr.3 und maximal 50 Zeichen
```

Soll sowohl nach Schrift als auch Schriftschnitt gesucht werden, so können beide Angaben durch & verknüpft werden, müssen aber in Anführungsstriche gesetzt werden.

```
bash-3.2$ luafindfont -i 4 "myriad & semibold"
We are using Lua 5.3
Looking for font "myriad & semibold"
Check for file /usr/local/texlive/2021/texmf-var/luatex-cache/generic/names/luaotfload-names
 No.                      Fontname Symbolic Name                          Path
   1.          MyriadPro-Semibold.otf myriadpro  /Users/voss/Library/Fonts/MyriadPro/
   2.      MyriadPro-SemiboldCond.otf myriadpro  /Users/voss/Library/Fonts/MyriadPro/
   3.    MyriadPro-SemiboldCondIt.otf myriadpro  /Users/voss/Library/Fonts/MyriadPro/
   4.        MyriadPro-SemiboldIt.otf myriadpro  /Users/voss/Library/Fonts/MyriadPro/
   5.    MyriadPro-SemiboldSemiCn.otf myriadpro  /Users/voss/Library/Fonts/MyriadPro/
   6.  MyriadPro-SemiboldSemiCnIt.otf myriadpro  /Users/voss/Library/Fonts/MyriadPro/
   7.   MyriadPro-SemiboldSemiExt.otf myriadpro  /Users/voss/Library/Fonts/MyriadPro/
   8. MyriadPro-SemiboldSemiExtIt.otf myriadpro  /Users/voss/Library/Fonts/MyriadPro/

Font: myriadpro

Fonttype otf(system) --› | Regular | Bold | Italic | BoldItalic |
```

Für das Programm `otfinfo` sind verschiedene Optionen möglich, wobei die Option `i` der Standard ist.

otfinfo

```
  s   Report font's supported scripts.
  f   Report font's GSUB/GPOS features.
  z   Report font's optical size information.
  p   Report font's PostScript name.
  a   Report font's family name.
  v   Report font's version information.
  i   Report font's names and designer/vendor info.
  g   Report font's glyph names.
  t   Report font's OpenType tables.
```

```
bash-3.2$ luafindfont -o 2f "myriad & semibold"
We are using Lua 5.3
Looking for font "myriad & semibold"
```

```
Check for file /usr/local/texlive/2021/texmf-var/luatex-cache/generic/names/luaotfload-
 ↪names
 No.                        Fontname Symbolic Name                                  Path
   1.          MyriadPro-Semibold.otf myriadpro  /Users/voss/Library/Fonts/MyriadPro/
   2.      MyriadPro-SemiboldCond.otf myriadpro  /Users/voss/Library/Fonts/MyriadPro/
   3.    MyriadPro-SemiboldCondIt.otf myriadpro  /Users/voss/Library/Fonts/MyriadPro/
   4.        MyriadPro-SemiboldIt.otf myriadpro  /Users/voss/Library/Fonts/MyriadPro/
   5.    MyriadPro-SemiboldSemiCn.otf myriadpro  /Users/voss/Library/Fonts/MyriadPro/
   6.  MyriadPro-SemiboldSemiCnIt.otf myriadpro  /Users/voss/Library/Fonts/MyriadPro/
   7.   MyriadPro-SemiboldSemiExt.otf myriadpro  /Users/voss/Library/Fonts/MyriadPro/
   8. MyriadPro-SemiboldSemiExtIt.otf myriadpro  /Users/voss/Library/Fonts/MyriadPro/

Run otfinfo -f: 2
otfinfo -f "/Users/voss/Library/Fonts/MyriadPro/MyriadPro-SemiboldCond.otf"
aalt  Access All Alternates
case  Case-Sensitive Forms
cpsp  Capital Spacing
dnom  Denominators
fina  Terminal Forms
frac  Fractions
kern  Kerning
liga  Standard Ligatures
lnum  Lining Figures
numr  Numerators
onum  Oldstyle Figures
ordn  Ordinals
pnum  Proportional Figures
sinf  Scientific Inferiors
sups  Superscript
tnum  Tabular Figures
zero  Slashed Zero
```

Schriftauswahl durch Namen

Die Auswahl einer Schrift durch ihren symbolischen Namen setzt voraus, dass die Schrift in einem der oben angegebenen Verzeichnisse zu finden ist. Dabei gehen XeLaTeX und LuaLaTeX unterschiedliche Wege; XeTeX sucht seine Schriften mit `fontconfig`, wozu die entsprechende Programbibliothek `https://www.freedesktop.org/wiki/Software/fontconfig/` Teil von XeTeX ist. Dagegen ermittelt LuaTeX die Schriften aus einem selbst erstellten Schriftenkatalog. Den »normalen« Anwender muss dies jedoch nicht sonderlich interessieren.

```
\fontspec{Cambria} Ein Test in der Schrift Cambria und nun ein Wechsel zur
{\fontspec{DejaVu Serif}[Scale=0.85] Schrift Dejavu und nun zu
{\fontspec{Arial}[Scale=0.9]allerletzt die Schrift Arial,}} die ebenso ein Klon
von \fontspec{HelveticaNeue}[Scale=0.9]Helvetica Neue ist, wie \TeX\ Gyre Heros.
```

Ein Test in der Schrift Cambria und nun ein Wechsel zur Schrift Dejavu und nun zu allerletzt die Schrift Arial, die ebenso ein Klon von Helvetica Neue ist, wie TEX Gyre Heros.

Die korrekten symbolischen Namen einer Schrift kann man, wie oben gezeigt, durch verschiedene Programme erfahren.

Schriftauswahl durch Dateinamen

Für XeTeX ist diese Variante immer dann zu benutzen, wenn man OpenType- oder TrueType-Schriften aus dem vorhandenen TeX-Baum benutzen möchte, die normalerweise dem zugrunde liegenden Betriebssystem formal nicht bekannt sind. Für LuaTeX besteht diese Einschränkung nicht; TeX-Bäume werden ebenfalls durchsucht.

```
\fontspec{Iwona} Ein Test in der Schrift Iwona und nun ein Wechsel zur
{\fontspec{Kurier}Schrift Kurier und nun zur Schrift {\fontspec{Antykwa Poltawskiego}
Antykwa Poltawskiego,}} die nicht wie \fontspec{HelveticaNeue}Helvetica Neue aussieht.
```

Ein Test in der Schrift Iwona und nun ein Wechsel zur Schrift Kurier und nun zur Schrift Antykwa Poltawskiego, die nicht wie Helvetica Neue aussieht.

Da LuaTeX seinen eigenen Cache für die Schriften verwaltet, kommt es beim ersten Aufruf zu einer Verzögerung des Übersetzungsvorgangs, da dieser Cache erst eingerichtet werden muss:

```
[...]
luaotfload | db : Font names database not found, generating new one.
luaotfload | db : This can take several minutes; please be patient.(compiling luc: /h
ome/voss/texlive/2016/texmf-var/luatex-cache/generic/fonts/otf/lmroman10-regular
.luc)(compiling luc: /home/voss/.texlive2016/texmf-var/luatex-cache/generic/font
s/otf/lmroman10-regular.luc)(save: /home/voss/texlive/2016/texmf-var/luatex-cach
[...]
```

Möchte man obiges Beispiel mit XeTeX laufen lassen, so wird es zu einer Fehlermeldung kommen, da die Schriften Iwona, Kurier und Antykwa Poltawskiego nicht von XeTeX gefunden werden, wenn nicht der Dateiname angegeben wird. Im folgenden Beispiel muss daher bei den ersten drei Schriften der Dateiname mit Endung angegeben werden, wohingegen die HelveticaNeue nach wie vor über den Namen geladen wird, da sie in der hier vorliegenden Version eine Systemschrift ist.

```
\fontspec{Iwona-Regular.otf} Ein Test in der Schrift Iwona und nun ein Wechsel
zur Schrift {\fontspec{Kurier-Regular.otf} Kurier und nun zu
{\fontspec{antpolt-regular.otf} allerletzt die Schrift Antykwa Poltawskiego,}} die
nicht wie \fontspec{HelveticaNeue} Helvetica aussieht.
```

Ein Test in der Schrift Iwona und nun ein Wechsel zur Schrift Kurier und nun zu allerletzt die Schrift Antykwa Poltawskiego, die nicht wie Helvetica aussieht.

Die Dateiendung kann man über das optionale Argument `Extension` und ein Verzeichnis, welches nicht im normalen Suchpfad liegt, über die Option `Path` vorgeben. Allerdings wird in diesem Fall nur der angegebene Schriftschnitt aktiviert; in folgendem Beispiel gibt daher `\textbf` keine fette Auszeichnung, da keine entsprechende fette Variante vereinbart, beziehungsweise nicht von `fontspec` gefunden wurde.

```
\setmainfont{BertholdWalbaumBook.ttf}Ein Test mit der Berthold-Walbaum-Schrift:\par
Ein ganz normaler Text in der alten schönen Schrift, die als \bfseries TrueType eingebunden
 ↪wurde.
```

Ein Test mit der Berthold-Walbaum-Schrift:
Ein ganz normaler Text in der alten schönen Schrift, die als TrueType eingebunden wurde.

Die Zuordnung zu der fetten Variante kann über die Option `BoldFont` erfolgen.

```
\setmainfont{BertholdWalbaumBook}[Path=fonts/,
  Extension=.ttf, BoldFont=BertholdWalbaumMediumBook]
Ein Test mit der Berthold-Walbaum-Schrift:\par Ein ganz normaler Text in der alten schönen
Schrift, die als \bfseries TrueType eingebunden wurde.
```

Ein Test mit der Berthold-Walbaum-Schrift:
Ein ganz normaler Text in der alten schönen Schrift, die als **TrueType eingebunden wurde.**

Die Eingabe der Schriften über einen Namen kann bei Anwendung von LuaLaTeX vereinfacht werden, wenn mit dem Platzhalter ∗ gearbeitet wird. Für den Basisnamen braucht dann nur der Teil angegeben zu werden, der für alle gleich ist. In folgendem Beispiel werden die Schriften nur über den Basisnamen BetholdImagoBQ gefunden, wobei dieser Basisname selbst noch kein Schriftname ist. Deshalb muss auch für `UprightFont` eine Definition erfolgen.

```
BertholdImagoBQ-Book.otf
BertholdImagoBQ-BookItalic.otf
BertholdImagoBQ-MediumItalic.otf
BertholdImagoBQ-Medium.otf
```

```
\setmainfont{BertholdImagoBQ}[%
  UprightFont=*-Book, ItalicFont=*-BookItalic,
  BoldItalicFont=*-MediumItalic, BoldFont=*-Medium]
Ein Test mit der Berthold-Imago-Schrift:\par  Ein ganz normaler Text in der neuen schönen
Schrift, die als \textbf{OpenType} eingebunden wurde.  \textit{Die Schrift als Italic und
nun zusätzlich \bfseries als fette Variante}
```

Ein Test mit der Berthold-Imago-Schrift:
Ein ganz normaler Text in der neuen schönen Schrift, die als **OpenType** eingebunden wurde. *Die Schrift als Italic und nun zusätzlich **als fette Variante***

Schriftfamilien

Mit den bisherigen Beispielen wurde jeweils die Hauptschrift festgelegt, was sich auch noch nachträglich durch weitere Aufrufe ändern lässt, wobei `\setromanfont` selten benutzt wird, da es im Allgemeinen der Hauptschrift entspricht. Die alte Syntax mit `\setmainfont[Optionen]{Schriftname}` ist aus Kompatibilitätsgründen für alle Makros weiterhin möglich.

Im Allgemeinen ist die Definition der Schriften mit ihren zugehörigen Features sehr aufwändig, wenn die Namensgebung der Schriften nicht so organisiert ist, dass das Paket `fontspec` die Zuordnung selbst vornehmen kann. Es existieren zur Zeit einige Pakete, die dem Anwender diese Arbeit abnehmen. Sie unterscheiden dabei bereits die verwendete TeX-Engine und laden selbstständig die benötigten Schriftformate. Obiges Beispiel kann daher vereinfacht werden:

```
\usepackage{Alegreya,AlegreyaSans}% lädt auch fontspec
\setmonofont{Anonymous Pro}[Scale=MatchLowercase]
```

```
Die normale Schrift ist die \textsc{Alegreya}, die es auch \textbf{fett gibt}.\par
\sffamily Die serifenlose die \textsc{Alegreya Sans}, die es auch \textbf{fett gibt}.\par
\ttfamily Und die Monoschrift ist die Anonymous Pro, die es auch \textbf{fett gibt}.\par
\addfontfeature{FakeStretch=0.65}%
Und die Monoschrift ist die Anonymous Pro, die es auch \textbf{fett gibt}.
```

Die normale Schrift ist die Alegreya, die es auch **fett gibt**.
Die serifenlose die Alegreya Sans, die es auch **fett gibt**.
Und die Monoschrift ist die Anonymous Pro, die es auch **fett gibt**.
Und die Monoschrift ist die Anonymous Pro, die es auch **fett gibt**.

Beispielhaft sei hier die vollständige Definition der Hauptschrift angegeben:

```
\setmainfont{Alegreya}%
  [ Numbers = {\Alegreya@figurealign,\Alegreya@figurestyle},
    UprightFont    = *-Regular ,
    ItalicFont     = *-Italic ,
    BoldFont       = *-\Alegreya@boldstyle ,
    BoldItalicFont = *-\Alegreya@boldstyle Italic ,
    UprightFeatures = { SmallCapsFont = AlegreyaSC-Regular },
    BoldFeatures = { SmallCapsFont= AlegreyaSC-\Alegreya@boldstyle },
    ItalicFeatures = { SmallCapsFont= AlegreyaSC-Italic },
    BoldItalicFeatures = {  SmallCapsFont= AlegreyaSC-\Alegreya@boldstyle Italic },
  ]
```

ConTEXt kurz notiert

Henning Hraban Ramm

> In der ConTEXt-Welt ist vieles in Bewegung, das eine Erwähnung wert ist, aber keinen ganzen Artikel begründet. (Stand: 22. 1. 2022)

Internationaler ConTEXt-»Stammtisch«

Seit Oktober 2021 gibt es ein regelmäßiges, englischsprachiges Online-Treffen für ConTEXt-Anwender, jeweils am zweiten Dienstag des Monats unter `https://lecture.senfcall.de/hen-rbr-rku-oke`, 15–17 Uhr MEZ. Der nachmittägliche Termin ist ein Kompromiss für die Teilnehmer aus aller Welt, von Singapur bis Kanada. Neben den allgemeinen Gesprächen über Projekte, Publishing, Programmieren und PDF ist das Treffen auch eine Sprechstunde mit Hans Hagen, der gerne tief in den Code taucht und Einzelheiten erklärt. Es ist erwünscht, Themen vorher auf der Mailingliste (NTG-CONTEXT) anzumelden; eventuelle Verschiebungen würden auch nur dort bekannt gegeben.

Fehlerverhalten

In allen TEX-Varianten ist das Verhalten im Fehlerfall schwierig. Aus LuaMetaTEX wurde schon die interaktive Textkonsole entfernt, weil die Engine danach ohnehin nicht sinnvoll weiterarbeiten kann, auch wenn z. B. ein Fehler in einer Quelldatei korrigiert wurde. Das Aufrufskript (`context` bzw. `mtxrun`) gab bisher aber nur in

wenigen Fällen einen Fehlercode zurück, was die Automatisierung von Arbeitsabläufen erschwerte. Dieses Verhalten wurde jetzt verbessert; mit `\enabledirectives[ ↪logs.errors=*]` oder dem Aufrufparameter `--errors='*'` gibt ConTEXt konsistente Fehlercodes zurück; welche Fehler so behandelt werden, lässt sich statt dem Jokerzeichen auch genau angeben. Für dieses Thema hat sich besonders Marco Patzer eingesetzt.

Bibliografie

Neben den Zitierregeln der APA (American Psychological Association), der APS (American Physical Society) und nach dem Chicago-Handbuch sind jetzt auch die Regeln der SBL (Society of Biblical Literature) implementiert. Das Projekt von Joey McCollum (`github.com/jjmccollum/context-sbl`) soll demnächst in die Distribution übernommen werden.

Präsentationen

Im `simpleslides`-Modul[1] von Aditya Mahajan und Thomas A. Schmitz wurden auf Initiative von Otared Kavian mehrere Fehler behoben, die sich im Laufe der Jahre angesammelt hatten. Es funktioniert jetzt vollständig mit ConTEXt LMTX.

Indische Sprachen

Durch den jüngsten Zustrom von ConTEXt-Anwendern, die indische Sprachen verwenden, ob muttersprachlich oder in wissenschaftlichen Arbeiten, ist Bewegung in diesen etwas vernachlässigten Bereich gekommen. Für die Silbentrennung von Sanskrit hatte sich früher offenbar niemand interessiert.

Manche Ligaturen werden von ConTEXt (MkIV und LMTX) gegenüber Harfbuzz (verwendet von LuaLaTEX und XeTEX) unterschiedlich dargestellt, in Abhängigkeit von der Schrift. Offenbar setzt Harfbuzz auf Heuristiken, die unterschiedlich implementierte OpenType-Features ausgleichen. Für ConTEXt müssen »Font Goodies« für die betroffenen Schriften erstellt werden. (Dieser Mechanismus wird sonst auch für fehlerhafte Schriften eingesetzt – diskutiert wurde in letzter Zeit z. B. die variable Version der EB Garamond von Google Fonts, in der gegenüber dem Original offenbar die Ligaturen durcheinander geraten sind.)

Kritische Ausgaben

Mehrere Anwender (Jean-Pierre Delange, Jürgen Hanneder, Pablo Rodriguez u. a.) möchten ConTEXt besser für historisch-kritische Ausgaben aufstellen bzw. die be-

[1] `github.com/adityam/simple-slides`

reits vorhandenen Mechanismen zusammentragen. Bisherige Vorstöße scheiterten meistens daran, dass die Anforderungen nicht genau genug spezifiziert wurden.

Thomas A. Schmitz (Altphilologe) führte aus, dass wissenschaftliche Publikationen ohne digitale Version heute unrealistisch seien. Der Weg von TEX zu XML (insbesondere TEI) sei in manchen Fällen möglich, aber wenig sinnvoll.

XML als Quellformat ist nicht besonders beliebt, aber ConTEXt wäre damit gut für die Erstellung einer Druckversion geeignet. Beispielsweise wird die italienische Werkausgabe von Marx und Engels (Projekt MEO von Massimiliano Farinella) mit Hilfe eines aufgebohrten HTML-Editors aufbereitet und mit ConTEXt gesetzt. Die Erstellung eines TEI-Regelsatzes für ConTEXt könnte also ein interessanter Ausgangspunkt für speziellere Projekte werden.

Jürgen Hanneder (Indologe) widersprach, dass er zumindest in seinem Fachbereich kaum TEI-basierte realisierte Editionsprojekte kenne und dass die XML-Eingabe die Arbeit stark verlangsame. Websites oder Web-Anwendungen seien zu flüchtig, da sie selten langfristig gepflegt würden, das zeige seine Erfahrung mit dem (TEI-basierten) »Nachtrags-Wörterbuch Sanskrit«[2]. Auch ein PDF zähle als digitale Veröffentlichung, und im Laufe der Diskussion habe sich ergeben, dass ConTEXt seine Anforderungen bereits erfülle, so dass kein eigenes Eingabeformat à la (re)ledmac erforderlich sei.

Unterstreichungen

Das Verhalten der Strich-Befehle (`\underbars`, `\underdot`, `\overbar` usw.) wurde verbessert, so dass sich auch unterschiedliche Striche stapeln bzw. verschachteln lassen.

Mathematiksatz

Dass der Formelsatz in ConTEXt konfigurierbarer geworden ist, habe ich schon in meinem letzten Beitrag erwähnt. Der schwedische Mathematiker Mikael Sundqvist arbeitet an einer Anleitung zum Formelsatz, in der er auch zeigt, wie bestimmte Vorgaben der AMS, der Uni Chicago und die eines schwedischen Verlags[3] umgesetzt werden können. Das hat dazu geführt, dass Hans Hagen den Zeilenumbruch langer Formeln verbessert hat und `\discretionary` jetzt auch im Mathe-Modus verfügbar ist – einer der Fälle, wo man sich fragt, »Warum war das nicht schon immer in TEX?« (Die Antwort lautet: »I Don't Knuth.«).

[2] `nws.uzi.uni-halle.de`

[3] W. N. Lansburgh, Almqvist & Wiksells Sättningsregler. 1961

Von fremden Bühnen

Präsentationen in XML[1]

Thomas A. Schmitz

XML ist ein Standard im Informationsaustausch. ConTEXt enthält ausgeklügelte und mächtige Werkzeuge zur Verarbeitung der vielseitigen Auszeichnungssprache. Diese Anleitung zeigt, wie man Präsentationen in XML definieren und dann mit ConTEXt verarbeiten kann.

XML verwenden

Grundprinzipien

XML bietet einen guten Kompromiss in Sachen Lesbarkeit für Menschen und Programme. Das Format ist ziemlich einfach, seine Prinzipien können in wenigen Sätzen beschrieben werden:

- XML-Dokumente bestehen aus Inhalt und Markup; wer bereits eine Markupsprache (wie TEX oder HTML) beherrscht, sollte keine großen Schwierigkeiten damit haben.
- Als Auszeichnungssprache muss XML ein paar Zeichen reservieren, um Auszeichnungen und Text unterscheiden zu können. In XML gibt es nur drei Arten von Sonderzeichen: spitze Klammern `<` `>` begrenzen ›Tags‹ (Elemente), die Zeichen `&` und `;` begrenzen Zeichen-›Entitys‹, und die geraden Anführungszeichen `"` und `'` begrenzen Attributwerte.
- XML-Dokumente sind (zumindest im Prinzip) in sich abgeschlossen: Sie beschreiben ihre eigene Struktur, d. h. sie definieren alle erlaubten Tags und deren erlaubten Inhalt.
- XML-Dokumente zeichnen sich durch eine hierarchische Struktur aus: Es gibt genau ein Wurzelelement (›Root‹); alle Elemente werden durch öffnende und schließende Tags begrenzt.

[1] Der Artikel erschien auf englisch im Context Group Journal zum ConTEXt meeting 2019; für Die TEXnische Komödie übersetzt und bearbeitet von Henning Hraban Ramm.

- Jedes öffnende Tag hat ein zugehöriges schließendes. Elemente müssen in der richtigen Reihenfolge verschachtelt werden und dürfen sich nicht überschneiden.

Im Gegensatz zu vielen anderen Markupsprachen definiert XML nur diese *syntaktischen* oder *strukturellen* Regeln, keine *semantischen*: Es gibt keine vordefinierten Tags und fast keine vordefinierten Entitys. Das hat den Vorteil, dass man seine eigenen Tags definieren und sie den Anforderungen des eigenen Dokuments anpassen kann. Der Nachteil ist, dass man seine eigenen Tags definieren muss und sie den Anforderungen des eigenen Dokuments anpassen muss. Das bedeutet, dass man sich über die Struktur seiner Informationen sorgfältig Gedanken machen sollte.

Sollte man XML verwenden?

XML bietet eine Reihe von Vorteilen, durch seine eigenen Qualitäten und durch die Art, wie es verwendet wird:

- Es ist extrem anpassungsfähig und kann für verschiedene Zwecke auf sehr unterschiedliche Weise verwendet werden. Es ist für stark strukturierte und verschachtelte Daten ebenso geeignet wie für »gemischte« Inhalte mit viel Text.
- Es ermutigt[2] die Trennung von Inhalt und Darstellung und damit das Schreiben von »sauberem« Code.
- XML kann von sehr vielen Programmen gelesen und verarbeitet werden. Fast jede Programmiersprache bietet Parser und Konverter dafür. Zusätzlich gibt es mächtige Werkzeuge für XML-basierte Workflows.
- Es gibt eine Reihe von Ausgabemöglichkeiten. Mit den richtigen Werkzeugen kann man PDFs zur Druckausgabe produzieren oder verschiedene Formate für das Web.
- XML bietet einen guten Kompromiss zwischen Lesbarkeit und Informationsdichte.
- Man könnte argumentieren, dass die etwas langatmige Syntax die Lesbarkeit erhöhe: Das explizite Öffnen und Schließen von Tags ist leichter verständlich als z. B. die geschweiften Klammern von TEX, bei denen man nicht immer gleich sieht, welche Gruppe sie schließen.

Das sind starke Argumente, aber XML ist keine Wunderwaffe und nicht per se ›besser‹ als TEX-Syntax.

Es gibt eine Reihe von Nachteilen, die man nicht vernachlässigen kann, wenn man XML mit ConTEXt verwendet:

- Man muss eine weitere Syntax lernen.

[2] Ich spreche von ›ermutigen‹, weil man auch in XML fürchterliche Dokumente erstellen kann, die optisches Erscheinungsbild und Struktur vermischen.

- XML fügt eine weitere Ebene zwischen Quelle und Ausgabe ein; man muss XML-Elemente auf die richtigen ConTEXt-Befehle abbilden.
- Das Schreiben von Makros wird verkompliziert: Man muss immer zweimal über die Syntax und Regeln nachdenken, zuerst, wie man sein Vorhaben in XML strukturiert, und dann, wie man die XML-Struktur in ConTEXt umsetzt.
- Es kann unheimlich frustrieren, wenn etwas nicht funktioniert: Die zusätzliche Übersetzung von XML nach TEX macht die Fehlersuche schwierig und ist oft eine Problemquelle.

Alles in allem empfehle ich, sich die Sache genau zu überlegen. Wer nur einzelne Dokumente verfasst, die sich in Struktur, Stil und Format unterscheiden, oder wer nur die PDF-Ausgabe braucht, sollte XML lieber vermeiden.

Es ist aber nützlich, wenn

- die Struktur der Dokumente sich vorhersehbar wiederholt,
- sich diese Struktur in wiederholten Stilelementen äußert,
- man verschiedene Ausgabeformate aus einer Quelle erzeugen möchte,
- man den Inhalt eventuell auf andere Weise oder mit anderen Anwendungen verarbeiten möchte.

XML mit ConTEXt verarbeiten

Der Editor

Die erste Frage, die viele Leute stellen, wenn sie von XML hören, ist: Welchen Editor sollte ich dafür verwenden? Darauf gibt es keine allgemeingültige Antwort, sie hängt von verschiedenen Faktoren ab: dem Betriebssystem, der Art der XML-Dokumente (enthalten sie hauptsächlich Text, oder sind es eher Datenbanken?) und der bevorzugten Arbeitsweise (möchte man die Tags und Attribute sehen oder sollen sie irgendwie interpretiert werden?).

Es gibt ein paar Spezialwerkzeuge zur Bearbeitung von XML. Das bekannteste ist Oxygen (`www.oxygenxml.com`), ein kommerzielles Programm, das für die meisten Plattformen verfügbar ist. Die meisten Universaleditoren bieten zumindest eine Syntaxhervorhebung.

Meine bevorzugte Lösung ist ziemlich einfach: Emacs ist mein Editor der Wahl, und es gibt dafür einen guten XML-Bearbeitungsmodus[3] mit Syntaxhervorhebung, Einrückung, Validierung und sogar automatischer Vervollständigung, wenn man ein Schema hinterlegt hat.

Man sollte einfach verschiedene Editoren ausprobieren, bis man etwas gefunden hat, mit dem man bequem arbeiten kann.

[3] `www.thaiopensource.com/nxml-mode`

XML in ConTEXt

XML mit ConTEXt zu verarbeiten ist schon lange möglich. Im alten ConTEXt MkII war das noch ziemlich kompliziert, aber mit ConTEXt MkIV, das auf LuaTEX basiert, hat sich die Unterstützung dramatisch verbessert, und das ist auch schon mehr als zehn Jahre her. Bei ConTEXt LMTX soll es noch besser werden.

Die XML-Datei wird in Lua geparst und im Speicher gehalten. Dadurch hat man jederzeit Zugriff auf jedes Element. Das macht es einfach, Teile der Eingabe mehrfach zu verwenden, auszuwählen, zu filtern, zu verändern oder zu prüfen. So kann man Makros schreiben, die das *n*-te Element des Typs ‹x› verarbeiten, oder ein Element des Typs ‹x› nur, wenn es den Text *z* enthält; oder man kann den Inhalt eines Elements mit verschiedenen Werten vergleichen und in Abhängigkeit davon etwas damit anstellen.

Um XML mit ConTEXt zu verarbeiten, braucht man neben der XML-Datei eine Stilvorlage (Environment), die an einem Ort liegen muss, wo ConTEXt sie findet (z. B. im gleichen Verzeichnis wie die XML-Datei). Sie enthält zwei Arten von Anweisungen:

- Einstellungen für jedes XML-Element, das man verarbeiten möchte, und
- Einstellungen für Layout und Aussehen des Dokuments – so wie bei jedem anderen ConTEXt-Projekt.

Wenn wir eine Datei `document.xml` und eine Stilvorlage `style.tex` haben, erzeugt der Aufruf:

```
context --environment=style document.xml
```

eine Datei namens `document.pdf`.

Präsentationen in XML

Um zu lernen, wie man XML in ConTEXt verarbeitet, gehen wir von einem realistischen Beispiel aus. Ich stelle den Arbeitsablauf vor, den ich nun seit 2017 an meiner Universität verwende.

Während des Semesters halte ich jede Woche eine 90-minütige Vorlesung. Ich schreibe den Text der Vorlesungen in XML, und ich habe mich entschieden, dass meine Quelldatei sowohl den Text meiner Vortragsfolien als auch mein Skript enthalten soll. Das hat den Vorteil, dass ich den Text mit anderen Anwendungen verarbeiten kann, um z. B. eine Website für den Kurs zu erzeugen, und dass ich meine Folien, Handouts für die Studierenden und das Skript für mich selbst aus den gleichen Quellen erzeugen kann, einfach mit unterschiedlichen Stilvorlagen.

Für die Präsentation verwende ich das `simpleslides`-Modul[4], das Aditya Mahajan und ich vor einigen Jahren entwickelt haben. Damit ist es einfach, Präsentationsfolien mit einer ansprechenden Gestaltung zu produzieren, ohne dass man sich allzusehr um Einstellungen und Layout kümmern muss; die ›Schwerarbeit‹ wird von dem Modul übernommen.

Die Struktur

Wie wir bereits gesehen haben, können wir die Elemente unserer XML-Datei selbst definieren. Was wäre dann eine gute Struktur für meinen Kurs? Die Datei soll die Präsentationen für ein ganzes Semester enthalten. Fangen wir mit dem Wurzelelement an, das wir etwas einfallslos `‹document›` nennen:

```
‹?xml version="1.0" encoding="UTF-8"?›
‹!DOCTYPE presentation›
‹document›

‹/document›
```

Dieses Wurzelelement bekommt Kindelemente namens `‹presentation›` für jede Vorlesung. Um jeder Präsentation einen eindeutigen Bezeichner zu geben, ergänzen wir ein Attribut `day` mit dem Datum der Vorlesung. Damit sieht die Struktur so aus:

```
‹document›

  ‹presentation day="21_09_21"›

  ‹/presentation›

  ‹presentation day="21_09_28"›

  ‹/presentation›

‹/document›
```

Jede Präsentation hat ein paar Metainformationen wie Titel und Datum, die auf der Titelfolie erscheinen sollen. Und natürlich hat sie einen Inhalt: die einzelnen Folien, aus denen die Präsentation besteht. Daraus ergibt sich die Struktur für unsere erste Präsentation:

```
‹document›

  ‹presentation day="21_09_21"›
```

[4] Die Installation ist im Wiki beschrieben: `https://wiki.contextgarden.net/Modules`

```
    <title>Beispiel einer Präsentation in XML</title>
    <author>A. U. Thor</author>
    <date>21. 9. 2021</date>
    <content>
      <slide>

      </slide>
      <slide>

      </slide>
    </content>
  </presentation>

</document>
```

Jede einzelne Folie hat drei Teile: einen Titel, einen Inhalt, der auf der Folie zu sehen ist, und einen Kommentar, der nicht auf die Folie soll, sondern ins Vorlesungsskript.

Der Inhalt der Folien kann natürlich irgendetwas sein, was man da haben möchte. Der Einfachheit halber beschränken wir uns hier auf ein paar häufige Standardelemente: normalen Text, eine Aufzählung, ein Bild und schließlich eine Tabelle. Wir verwenden diese hier nur einzeln, damit die Beispiele nicht zu komplex werden.

Aber zunächst beschäftigen wir uns damit, wie man eine Stildatei zur Verarbeitung dieser XML-Elemente verfasst:

Die Stilvorlage

Unsere Datei `style.tex` beginnt mit folgenden Zeilen:

```
\startxmlsetups xml:presentationsetups
  \xmlsetsetup{#1}{*}{-}
  \xmlsetsetup{#1}{document|
                   presentation|
                   title|
                   author|
                   date|
                   content|
                   slide|
                   slidetitle|
                   slidecontent}
              {xml:*}
\stopxmlsetups

\xmlregistersetup
```

```
  {xml:presentationsetups}
```

Wir definieren und registrieren ein ›Setup‹ für unsere XML-Verarbeitungsanweisungen namens `presentationsetups`. Unsere Datei enthält zwar den Text für Folien *und* Skript, aber für die Präsentation wollen wir nur die Inhalte der Folien verarbeiten. Dazu müssen wir genau definieren, welche Elemente verwendet werden sollen. Zuerst weisen wir ConTEXt an, alle Elemente fallenzulassen; das macht die Anweisung `\xmlsetsetup{#1}{*}{-}`. Danach listen wir die Elemente auf, die wir verarbeiten wollen. Später müssen wir jedes neue Element zu dieser Liste hinzufügen. Die Liste enthält jetzt also alle bisher bekannten Elemente außer `‹slidecomment›`, dessen Inhalt wir ja nicht auf den Folien haben wollen.

Die Anweisung für das Wurzelelement `‹document›` ist einfach: ConTEXt soll den Inhalt einfach zur weiteren Verarbeitung ›spülen‹ (flush):

```
\startxmlsetups xml:document
  \xmlflush{#1}
\stopxmlsetups
```

Dies ist der erste und wichtigste Befehl, den man zur Verarbeitung von XML lernen muss: `\xmlflush{#1}` bedeutet »nimm den Inhalt des aktuellen Elements und verarbeite ihn«. Unter ›verarbeiten‹ verstehen wir dabei: Wenn es Text ist, setze ihn; wenn weitere XML-Elemente enthalten sind, verarbeite sie laut den zugehörigen Anweisungen.

Was machen wir mit den einzelnen Präsentationen? In unserem Beispiel sind sie mit einem Datum bezeichnet, und wir möchten eine einzelne Präsentation für jede Vorlesung bekommen. Wie erreichen wir das? Ich schlage vor, wir verwenden das Attribut `day` als Namen für einen ConTEXt-Modus:

```
\startxmlsetups xml:presentation
  \startmode[\xmlatt{#1}{day}]
  \setupTitle[
    title={\xmltext{#1}{title}},
    author={\xmltext{#1}{author}},
    date={\xmltext{#1}{date}}]
  \placeTitle
  \xmltext{#1}{content}
  \page
  \stopmode
\stopxmlsetups
```

Zuerst ziehen wir den Wert aus dem `day`-Attribut (das macht `\xmlatt{#1}{day}` und starten einen `mode` damit. Für unsere Präsentation mit dem Attributwert `"21_09_21"`

definiert das einen Modus dieses Namens, den wir dann auf der Kommandozeile verwenden können, um nur die Präsentation dieses einen Tages zu erzeugen:

```
context --environment=style --mode=21_09_21 document.xml
```

Dann sehen wir uns die Kindelemente unseres ‹presentation›-Elements an:

`\xmltext{#1}{title}` holt den Inhalt des ‹title›-Elements, der dann dem Befehl `\setupTitle` aus dem `simpleslides`-Modul als Parameter übergeben wird. Auf diese Weise werden Titel, Autor und Datum verarbeitet; der Befehl `\placetitle` erzeugt unsere Titelfolie. Schließlich überreichen wir das ‹content›-Element der Satzmaschine zur weiteren Behandlung.

Das war die allgemeine Struktur unserer Präsentation. Betrachten wir nun die einzelnen Folien. Die erste bekommt nur einen Titel und etwas Text. So könnte das XML dazu aussehen:

```
<slide>
  <slidetitle>Grundlagen</slidetitle>
  <slidecontent>
    ConTeXt ist eine Markupsprache und ein Verarbeitungssystem für Dokumente,
    basierend auf dem Satzsystem \TeX{}. Es wurde mit dem gleichen Ziel der
    allgemeinen Verwendbarkeit entwickelt wie LaTeX, um einen vereinfachten
    Zugang zum hochqualitativen Textsatz zu ermöglichen, den TeX bietet.
    Während LaTeX aber die Schreibenden von Layout und typografischen Details
    fernhält, nimmt ConTeXt den entgegengesetzten Weg, indem es strukturierte
    Schnittstellen zu Typographie, Hintergründen, Hyperlinks, Darstellung,
    Farben und bedingter Ausführung anbietet.
  </slidecontent>
  <slidecomment>
    Hier stehen ein paar Kommentare zu dieser Folie.
    Sie erscheinen nicht in der Präsentation, sondern
    nur im Vortragsskript.
  </slidecomment>
</slide>
```

Das ist nicht besonders kompliziert; wir müssen nur Regeln schreiben, die den Inhalt der Elemente ‹slidetitle› und ‹slidecontent› an ConTEXt übergeben:

```
\startxmlsetups xml:slide
  \xmldoiftext{#1}{/slidetitle}{%
  \SlideTitle{
    \xmltext{#1}{slidetitle}%
  }}
  \start
    \xmltext{#1}{slidecontent}
    \par
```

```
  \stop
\stopxmlsetups
```

In der ersten Zeile des Setups prüfen wir, ob das Element ‹slidetitle› überhaupt einen Inhalt hat, schließlich haben nicht alle Folien einen Titel; wenn ja, dann wird er mit dem Befehl \SlideTitle aus dem simpleslides-Modul gesetzt.

Der Inhalt der Folie steht in einer \start…\stop-Umgebung, so dass alle Schriftveränderungen innerhalb der Folie sich nicht darüber hinaus auswirken.

Der Inhalt der Folie wird dann wiederum einfach an ConTEXt zum Setzen weitergegeben. Für unsere einfache Folie reicht das.

Als nächstes kommt eine Folie mit einer Aufzählung. Hier ist das XML:

```
‹slide›
  ‹slidetitle›Eine Aufzählung‹/slidetitle›
  ‹slidecontent›
    ‹numberedlist›
      ‹item›Erster Punkt‹/item›
      ‹item›Zweiter Punkt‹/item›
      ‹item›Dritter Punkt‹/item›
      ‹item›Vierter Punkt‹/item›
    ‹/numberedlist›
  ‹/slidecontent›
  ‹slidecomment›
    Mehr Anmerkungen.
  ‹/slidecomment›
‹/slide›
```

Für diesen Fall müssen wir die Einstellungen für zwei neue XML-Elemente definieren, ‹numberedlist› und ‹item› – und nicht vergessen, diese Namen in die Elemente-Liste am Anfang der Stilvorlage einzutragen.

Für die Aufzählung verwenden wir einfach eine itemize-Umgebung:

```
\startxmlsetups xml:numberedlist
  \startitemize[n]
    \xmlflush{#1}
  \stopitemize
\stopxmlsetups

\startxmlsetups xml:item
  \startitem
    \ignorespaces\xmlflush{#1}
  \stopitem
\stopxmlsetups
```

Zuerst geben wir an, dass ConTEXt den Inhalt des Elements ‹numberedlist› innerhalb einer itemize[n]-Umgebung »flushen« soll, und dann jedes einzelne ‹item› in einer item-Umgebung.

Unsere nächste Folie soll ein Bild zeigen. Das simpleslides-Modul enthält bereits Code, um Bilder ansprechend einzubinden, also müssen wir das nur noch in XML abbilden und an ConTEXt weitergeben. So könnte das aussehen:

```
‹slide›
  ‹slidecontent›
    ‹includeimage type="vertical"
    resource="cow" height="0.4"›
      Bildunterschrift
    ‹/includeimage›
  ‹/slidecontent›
  ‹slidecomment›
    Muh!
  ‹/slidecomment›
‹/slide›
```

Und hier ist das Setup für die XML-Struktur:

```
\startxmlsetups xml:includeimage
  \IncludePicture[\xmlatt{#1}{type}]
    [\xmlatt{#1}{resource}]
    [height=\xmlatt{#1}{height}\textheight]
  {\xmlflush{#1}}
\stopxmlsetups
```

Wir erinnern uns: \xmlatt{#1}{type} bedeutet »der Wert des Attributs type des aktuellen XML-Elements«. Ein XML-Element kann mehrere Attribute haben, und mit diesem ConTEXt-Befehl können wir sie abfragen. Oh, und nicht vergessen, das neue Element ‹includeimage› in die Liste am Anfang der Stilvorlage einzutragen!

Schließlich hätten wir gerne eine Tabelle auf einer unserer Folien. Hierfür benutzen wir »Extreme Tables«, obwohl »Natural Tables« (alias »HTML-Tabellen«) ebensogut funktionieren würden. Um es ein bisschen anspruchsvoller zu machen, verwenden wir auch spalten- oder zeilenübergreifende Zellen.

So drücken wir das in XML aus:

```
‹slide›
  ‹slidetitle›EIne Tabelle‹/slidetitle›
  ‹slidecontent›
  ‹table›
    ‹tablerow›
```

```
      <tablecell>Habt</tablecell>
      <tablecell>ihr</tablecell>
      <tablecell>schon</tablecell>
      <tablecell>gesehen,</tablecell>
     </tablerow>
     <tablerow>
     <tablecell>was</tablecell>
     <tablecell nx="2" ny="2">
       &CONTEXT;
     </tablecell>
     <tablecell>für</tablecell>
    </tablerow>
    <tablerow>
      <tablecell>all</tablecell>
      <tablecell>unsere</tablecell>
    </tablerow>
    <tablerow>
      <tablecell>wunderbaren</tablecell>
      <tablecell>Dokumente</tablecell>
      <tablecell>tun</tablecell>
      <tablecell>kann?</tablecell>
    </tablerow>
  </table>
  </slidecontent>
  <slidecomment>
    Noch mehr Anmerkungen.
  </slidecomment>
</slide>
```

Die Abbildung auf ConTEXt-Befehle ist ziemlich simpel:

```
\startxmlsetups xml:table
  \placefigure[here,force]{none}
   {\startembeddedxtable
    \xmlflush{#1}
  \stopembeddedxtable}
\stopxmlsetups

\startxmlsetups xml:tablerow
  \startxrow
    \xmlflush{#1}
  \stopxrow
\stopxmlsetups
```

```
\startxmlsetups xml:tablecell
  \startxcell[
    nx=\xmlattdef{#1}{nx}{1},
    ny=\xmlattdef{#1}{ny}{1},
    align=middle,
    top=\vss,
    bottom=\vss]
    \xmlflush{#1}
  \stopxcell
\stopxmlsetups
```

Hier sehen wir eine andere Methode, um XML-Attribute zu verarbeiten: `\xmlattdef ↪{\#1}{nx}{1}` bedeutet: »der Wert des Attributs `nx` des aktuellen Elements; wenn es das nicht gibt, nimm den Vorgabewert ›1‹«.

Außerdem haben wir eine XML-Entity verwendet: Wir wollen das Logo ConTEXt ordentlich gesetzt haben, also haben wir es in XML als `&CONTEXT;` ausgedrückt. Dafür brauchen wir eine Definition in unserer Stildatei:

```
\xmltexentity{CONTEXT}{\ConTeXt}
```

Fast geschafft!

Unsere Stilvorlage muss nun das Aussehen unseres Dokuments definieren. Wir machen es uns einfach und überlassen das Meiste dem `simpleslides`-Modul:

```
\usemodule[simpleslides][
  style=BigNumber,
  font=Helvetica,
  size=17pt]
```

Wenn wir unser XML nun mit dem bekannten Aufruf compilieren:

```
context --environment=style --mode=11_09_21 document.xml
```

bekommen wir eine Ausgabe, die aussieht wie Abb. 1 auf S. 72.

Das Skript

Das waren soweit die Einstellungen, um eine PDF-Präsentation aus unserer XML-Datei zu machen. Wir betrachten nun die Möglichkeiten, aus den gleichen Daten eine andere Ausgabe zu erzeugen.

Unsere erste Übung ist das Vorlesungsskript. Dabei wollen wir gewissermaßen das Gegenteil der Präsentation erreichen: Wir wollen nur den Inhalt der `‹slidecomment›`-Elemente und alles andere ignorieren. Nachdem wir (bisher noch) keine besonderen

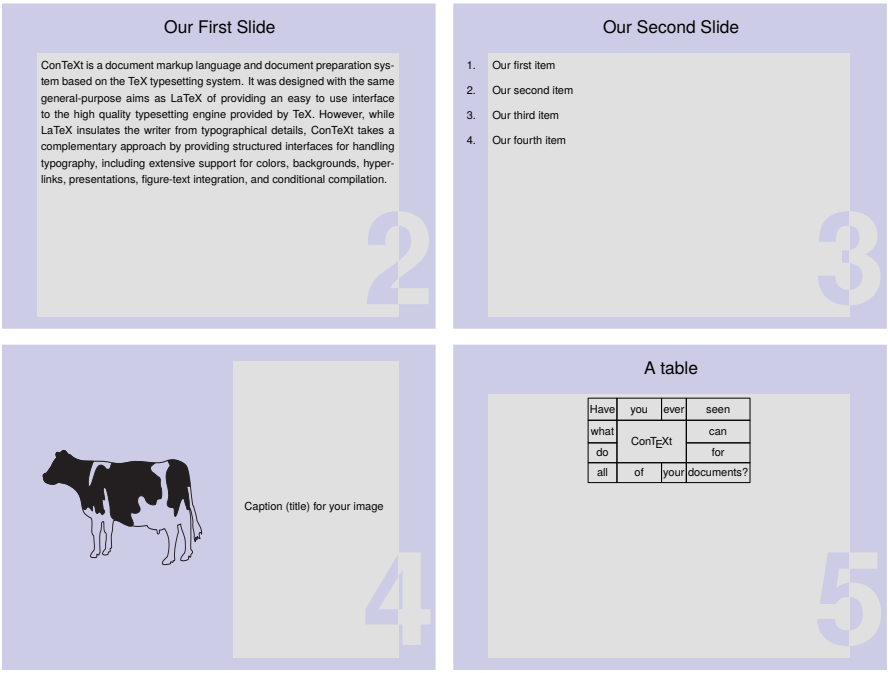


Abb. 1: Ausgabe unserer Beispielpräsentation (englische Version)

Elemente im Kommentarbereich verwenden, müssen wir nur dafür sorgen, dass der Inhalt gesetzt wird.

Als auffälliges Element fügen wir einen Folienzähler ein, damit wir nicht den Überblick verlieren, wo in unserer Präsentation wir uns gerade befinden.

Nachdem wir das Grundprinzip der XML-Verarbeitung verstanden haben, können wir uns die zweite Stilvorlage einfach so ansehen:

```
\startxmlsetups xml:manuscriptsetups
\xmlsetsetup{#1}{*}{-}
\xmlsetsetup{#1}{document|
                 presentation|
                 content|
                 slide|
                 slidecomment}{xml:*}
\stopxmlsetups

\xmlregistersetup{xml:manuscriptsetups}
```

```
\startxmlsetups xml:document
  \xmlflush{#1}
\stopxmlsetups

\startxmlsetups xml:content
  \xmlflush{#1}
\stopxmlsetups

\startxmlsetups xml:presentation
  \startsection[
    title={\xmltext{#1}{date}: \hfill
    \xmltext{#1}{title}},
    bookmark={\xmltext{#1}{date}:\xmltext{#1}{title}}]
    \xmlflush{#1}
  \stopxmlsetups

\startxmlsetups xml:slide
  \NewSlide \xmlflush{#1}
\stopxmlsetups

\startxmlsetups xml:slidecomment
  \xmlflush{#1}\par
\stopxmlsetups

\definecounter [SlideNumber] [way=bytext,prefix=no]

\setuplayout[
  marking=off,
  width=fit,
  height=fit,
  header=0.6cm,
  footer=0cm]

\setuphead[section][
  style=normal,
  number=yes,
  after={\resetcounter[SlideNumber]},
  expansion=yes,
  page=yes]

\setuppapersize[A6,landscape][A6,landscape]

\define\NewSlide%
```

```
  {\incrementcounter[SlideNumber]%
  \color[red]{[{\rawcounter[SlideNumber]}]}}

\setupinteraction[
  state=start,
  title={XML-Präsentationen},
  author={A. U. Thor}]

\placebookmarks[section][all]

\setupuserpagenumber[state=start,way=bysection]

\setupheadertexts[{\getmarking[sectionnumber]} – \pagenumber ]
```

Das meiste dürfte inzwischen ziemlich offensichtlich sein: Wir ›flushen‹ die Elemente, die wir gesetzt haben wollen. In diesem Fall möchten wir ein PDF für den gesamten Kurs, mit den einzelnen Präsentationen als Abschnitte (sections). Diese Abschnitte beginnen auf einer neuen Seite und verwenden das ‹title›-Element als Titel. Wir fügen außerdem Lesezeichen (bookmarks) in unser PDF ein, damit sich die einzelnen Vorlesungen leichter finden lassen.

Wir haben einen Zähler `\SlideNumber`, der für jede Folie inkrementiert und in rot ausgegeben wird. Die Kopfzeile enthält die Nummer der Vorlesung im Kurs und die Seitenzahl innerhalb der Präsentation.

Der Rest des Codes dient der Optik des Skriptes, das im A6-Querformat ausgegeben wird – entweder zum Ausdruck auf Karteikarten oder zur Anzeige auf einem Tablet.

Handouts

Ursprünglich habe ich meine Folien genau so zum Download bereitgestellt, wie ich sie im Hörsaal gezeigt habe (siehe Abb. 1). Die Studierenden beschwerten sich aber zu Recht, dass dieses Format nicht zum Ausdrucken geeignet war. Also habe ich ein Environment definiert, das den Inhalt der Folien ohne farbigen Hintergrund ausgibt und jeweils vier Folien untereinander auf der linken Seite einer A4-Seite abbildet, damit auf der rechten Seite Platz für Notizen bleibt. So können die Studierenden ihre Anmerkungen auf den ausgedruckten Handouts direkt zu den Folien schreiben.

Die meisten Einstellungen für die Elemente sind die gleichen wie für die Folien. Da wir aber ohne das `simpleslides`-Modul arbeiten, musste ich ein paar Definitionen aus dem Modul kopieren, z. B. für die Bildplatzierung. Aber damit will ich niemanden langweilen. Das Wichtigste ist die Anordnung der Folien auf der Seite, die wir mit folgendem Code erreichen:

```
\setuppapersize[A7,landscape][A4]

\setuppaper[
  topspace=3mm,
  backspace=1.5mm,
  bottomspace=0mm,
  dx=0mm,
  dy=0mm,
  nx=1,
  ny=4]

\setuparranging[XY]
```

Damit bekommen wir die gewünschte Anordnung für den Ausdruck.

Fazit

Es dauert eine Weile, einen Workflow aufzusetzen, mit dem man Präsentationen aus XML erzeugen kann, und ich habe mehrere Anläufe gebraucht, um zu einer Form zu kommen, die mir hoffentlich auch in Zukunft dienen wird.

Wenn man alles beisammen hat, werden die Vorteile des XML-Formats deutlich: Es ist bequem, sowohl die Folien als auch die Notizen in einer Datei zu haben und die Daten damit einfach wiederverwenden zu können. Das Material steht damit auch für andere Ausgabeformate zur Verfügung; es wäre nicht schwierig, die Folien mit Hilfe einer XSL-Transformation ins Netz zu bringen. Und nachdem Inhalt und Form sauber getrennt sind, sollten die Daten auch noch verwendet werden können, wenn sich die zugrundeliegenden Mechanismen ändern.

LaTeX News – Issue 34, November 2021[1]

Frank Mittelbach

Einleitung

Dieses LaTeX-Release enthält keine großen neuen Module. Stattdessen konzentriert es sich darauf, die in vorherigen Releases eingeführte Funktionalität zu konsolidieren und zu verbessern. Zudem wurden der Kernel und die zentralen Pakete um eine Reihe von kleineren Verbesserungen und Fehlerkorrekturen ergänzt.

[1] Der Newsletter wurde zuvor im *TUGboat* 42:3 [3] veröffentlicht und von Thomas Demmig übersetzt.

Hooks

Seit der Einführung des Hook-Management-Systems im 2020er Release von LaTeX [5] haben mehr und mehr Paket-Entwickler damit begonnen, diese Funktionalität einzusetzen. Ein Ergebnis dieser zunehmenden Aktivitäten ist die Anzahl an Fragen, die zeigte, dass ein Teil der Dokumentation nicht präzise genug war und manche Verbesserungen notwendig waren – diese Punkte wurden nun in der Dokumentation angegangen. Die zunehmende Verwendung hat auch ein kleine Zahl von (konzeptionellen) Fehlern aufgedeckt, die unserer Meinung nach korrigiert werden sollten, solange die Verbreitung noch relativ gering ist – die nachfolgenden Probleme wurden daher in diesem Release angegangen.

`\ActivateGenericHook` bereitstellen

Das Hook-Management-System bietet eine Reihe generischer Hooks, also solche, deren Namen eine variable Komponente wie zum Beispiel den Namen einer Umgebung enthalten. Es ist nicht praktikabel, solche Hooks vorzudefinieren, daher nutzen diese Hooks einen anderen Mechanismus: Sie sind implizit verfügbar und werden in dem Moment zum Leben erweckt, in dem ein Paket – oder die Präambel des Dokuments – einem von ihnen mit Hilfe von `\AddToHook` Code zuweist. Der Kernel bietet solche Hooks für Umgebungen (`env/...`) und Befehle an (`cmd/...`), zudem für Dateien, Pakete und Klassen (`file/...`, `include/...`, `package/...`, `class/...`).

Es ist auch möglich, solche generischen Hooks in Paketen anzubieten, wenn beispielsweise Hooks benötigt werden, die von der aktuellen Sprache abhängen und daher den Sprachnamen als Teil des Hook-Namens erfordern (und Sie im Voraus vermutlich nicht alle benötigten Namen kennen).

Wollen Sie solche generischen Hooks anbieten, können Sie dies nun mit `\UseHook` oder `\UseOneTimeHook` in Ihrem (Paket-)Code erreichen, dürfen dann aber *den Hook nicht mit `\NewHook` deklarieren.* Allerdings wird ohne weitere Aktivitäten ein Aufruf von `\UseHook` mit einem nicht deklarierten Hook auch nichts auslösen – als zusätzlicher Setup-Schritt ist es daher erforderlich, den generischen Hook explizit durch `\ActivateGenericHook` zu aktivieren.[2]

Unter der Annahme, dass Sie die am Ende verwendeten Hook-Namen nicht alle im Voraus kennen, bleibt es dem Anwender oder der Anwenderin Ihres Pakets überlassen, den Hook selbst zu aktivieren, bevor ihm Code hinzugefügt wird. So bietet beispielsweise Babel Hooks wie `babel/`⟨*language*⟩`/afterextras` an, die es den

[2] Beachten Sie, dass wir im vorigen Release `\ProvideHook` angeboten haben, um diesen Effekt zu erreichen, aber der Name war eine schlechte Wahl, daher entschieden wir uns dafür, ihn zu verwerfen und nun stattdessen `\ActivateGenericHook` einzuführen.

Anwendern ermöglichen, sprachspezifische Deklarationen beim Wechsel in eine Sprache hinzuzufügen. Man kann dann schreiben:

```
\ActivateGenericHook{babel/ngerman/afterextras}
\AddToHook{babel/ngerman/afterextras}{\color{blue}}
```

Dann würden alle deutschen Wörter im Text blau markiert werden.[3]

Beachten Sie, dass ein auf diesem Weg erzeugter generischer Hook immer ein normaler Hook ist.

Standardisierte Namen für die generischen Hooks

Der initiale Satz an generischen Hooks, der vom Kernel bereitgestellt wurde, baute auf zwei Namensmustern auf: Entweder hatten die Namen die Form `env/`⟨*name*⟩`/after`, bei der sich die Variable ⟨*name*⟩ in der Mitte befand, oder sie lauteten wie `file/after/`⟨*name*⟩, so dass der variable Teil an dritter Stelle stand. Die Koexistenz dieser beiden Muster sorgte für Verwirrung, weil man wissen musste, an welcher Position sich der variable Teil befindet – zudem wurde der Code komplizierter und langsamer.

Die mit Dateien in Bezug stehenden Hooks wurden daher umbenannt, so dass sich der variable Teil wie bei all den anderen Hooks an zweiter Stelle befindet, siehe Tabelle 1. Da diese Änderung nicht abwärtskompatibel ist, werden die alten

alter Name	neuer Name
`file/before/`⟨*name*⟩	→ `file/`⟨*name*⟩`/before`
`file/after/`⟨*name*⟩	→ `file/`⟨*name*⟩`/after`
`package/before/`⟨*name*⟩	→ `package/`⟨*name*⟩`/before`
`package/after/`⟨*name*⟩	→ `package/`⟨*name*⟩`/after`
`class/before/`⟨*name*⟩	→ `class/`⟨*name*⟩`/before`
`class/after/`⟨*name*⟩	→ `class/`⟨*name*⟩`/after`
`include/before/`⟨*name*⟩	→ `include/`⟨*name*⟩`/before`
`include/end/`⟨*name*⟩	→ `include/`⟨*name*⟩`/end`
`include/after/`⟨*name*⟩	→ `include/`⟨*name*⟩`/after`

Tab. 1: Änderung der Datei-Hook-Namen

Namen noch eine Weile funktionieren, so dass Paket-Autoren und Anwender genug Zeit haben, sich anzupassen – es wird aber eine Warnung ausgegeben, wenn die alten Namen zum Einsatz kommen. Später werden die veralteten Namen für Fehlermeldungen sorgen und dann vollständig entfernt werden. *(Github Issue 648)*

[3] Die Rückkehr zu Schwarz erfolgt dabei implizit, solange man `\foreignlanguage` verwendet. Schachtelt man die Befehle oder verwendet `\selectlanguage`, reicht das nicht, da die anderen Sprachen nicht explizit nach Schwarz wechseln – dazu müsste man ähnliche Definitionen auch für diese tätigen.

Einige Datei-Hooks wurden zu Einmal-Hooks gemacht

Klassen, Pakete und Include-Dateien können in einem LaTeX-Dokument nur einmal geladen werden. Aus diesem Grund wurden die Hooks, die spezifisch mit dem Laden solcher Dateien verbunden sind, zu Einmal-Hooks gemacht. So sind sie nicht nur effizienter, sondern es wird auch folgender wichtiger Anwendungsfall unterstützt:

```
AddToHook{package/varioref/after}{... beim Laden des Pakets anwenden oder sofort
                                    anwenden (wenn Paket schon geladen ist) ...}
```

Es muss dann nicht zuerst geprüft werden, ob das Paket schon geladen wurde. *(Github Issue 623)*

Zusätzlichen Hook-Code für den nächsten Aufruf aufräumen

Es gibt ein paar Anwendungsfälle, in denen es hilfreich ist, wenn man eine vorherige Verwendung von `\AddToHookNext` abbrechen könnte – zum Beispiel, wenn eine Seite mit `\DiscardShipoutBox` verworfen wurde, weil nur ein paar Seiten des Dokuments gedruckt werden. Für solche Situationen wird nun der neue Befehl `\ClearHookNext` angeboten. *(Github Issue 565)*

Nach `\UseOneTimeHook` aufräumen

Manche Hooks sind dazu gedacht, in einem Dokument nur einmal genutzt zu werden, und weitere Versuche, sie um Code zu ergänzen, sorgen dafür, dass der zusätzliche Code sofort ausgeführt wird, statt ihn zum Hook hinzuzufügen. Die erste Implementierung dieses Konzepts war sehr einfach und sie berücksichtigte nicht, dass Pakete versuchen könnten, einen Einmal-Hook mehrfach aufzurufen, so dass der Hook-Code auch wiederholt ausgeführt würde. Auch wenn die Implementierung für einfache Fälle ausreichend war (wie zum Beispiel den Hook `begindocument`), sorgte sie für Ärger, wenn der Einmal-Hook zum Beispiel als Initialisierungs-Hook gedacht war, der nur einmal verwendet wird (wenn ein Befehl erstmals aufgerufen wird), in weiteren Aufrufen aber zu ignorieren ist.

Dieser Nachteil wurde angegangen und jetzt wird ein Einmal-Hook wirklich nur einmal ausgeführt und der Code danach entfernt, um wieder etwas Speicher freizugeben. *(Github Issue 565)*

`\RemoveFromHook` mit einem fehlenden Code-Label

Gab es bei einem Aufruf der ersten Version von `\RemoveFromHook` das zu entfernende Code-Label nicht, wurde eine »Removal Order« in die Queue gestellt – beim nächsten Versuch, dieses Label zum Hook hinzuzufügen, wurde diese `\AddToHook`-Aktion

dann abgebrochen und kein Code hinzugefügt. So hätte im Prinzip die Reihenfolge beim Laden der Pakete keine Auswirkungen. Aber diese Implementierung funktionierte nicht wie gewünscht, denn während zwei `\AddToHook`-Aktionen mit einem bestimmten Label durch ein einzelnes `\RemoveFromHook` entfernt werden konnten, war ein `\RemoveFromHook` nicht dazu in der Lage, zwei `\AddToHook`-Aktionen für dieses Label abzubrechen – das sorgte für Verwirrung und führte zudem zu weiteren Problemen.

Die Implementierung wurde nun angepasst, so dass `\RemoveFromHook` nur Code-Label entfernt, die in einem Hook schon existieren: Existiert solch ein Code-Label nicht, wird eine Warnung ausgegeben.

Beachten Sie allerdings: Innerhalb eines Pakets sollten Sie `\RemoveFromHook` verwenden, um ein Code-Label zu entfernen, während man zum Entfernen von Code aus Hooks von anderen Paketen die `voids`-Relation benutzen sollte. Die Vorteile hierbei sind, dass es sich um eine nicht-destruktive Relation handelt (sie kann also später durch das Verwenden einer anderen Relation wieder entfernt werden) und sie zudem vollständig unabhängig von der Reihenfolge beim Laden der Pakete ist. *(Github Issue 625)*

Befehle mit Parameter-Token patchen

Im letzten Release wurde der Hook-Mechanismus von LaTeX so erweitert, dass er das Einschleusen von Code in beliebige Befehle mit Hilfe generischer `cmd`-Hooks ermöglicht.

Diese Version der Erweiterung besaß allerdings einen Fehler – das Patchen mancher Befehle mit einem Parameter-Token (normalerweise `#`) in ihrer Definition schlug mit einem Low-Level-TeX-Fehler fehl. Das wurde nun korrigiert, so dass ein Patchen für diese Befehle ebenfalls funktioniert. *(Github Issue 697)*

Neue oder verbesserte Befehle

`\NewCommandCopy` und `\ShowCommand` erweitert

Seit dem Release 2020-10-01 (siehe [5]) bietet LaTeX `\NewCommandCopy`, um robuste Befehle zu kopieren, und `\ShowCommand`, um deren Definitionen am Terminal auszugeben. Im gleichen Release wurde das Paket `xparse` in den Kernel integriert (als `ltcmd`), um `\NewDocumentCommand` und so weiter bereitzustellen. Aber die erweiterte Unterstützung für `\NewCommandCopy` und `\ShowCommand` wurde in `ltcmd` nicht sofort umgesetzt.

Das aktuelle LaTeX-Release implementiert jetzt diese Unterstützung, so dass nun auch mit `\NewDocumentCommand` und seinen Varianten definierte Befehle kopiert und

deren Definitionen einfach am Terminal angezeigt werden können, ohne \csname-»Spielchen« treiben zu müssen. *(Github Issue 569)*

Allozieren von Mathematik-Alphabeten bei Bedarf zurücknehmen

TEX – oder genauer gesagt die 8-Bit-Versionen von TEX, wie zum Beispiel pdfTEX – besitzen eine feste Grenze von maximal 16 verschiedenen Mathematik-Font-Gruppen (\fam oder \mathgroup), die in einer einzelnen Formel verwendet werden können. Für jeden (durch ein Paket oder in der Präambel) deklarierten Symbol-Font wird eine eigene Mathematik-Gruppe alloziert und das Gleiche geschieht auch für jedes Mathematik-Alphabet (wie beispielsweise \mathbf), sobald es irgendwo im Dokument zum Einsatz kommt. Bisher war die Allokation dieser Mathematik-Alphabete permanent, auch wenn sie nur einmalig genutzt wurden – das führte dazu, dass Ihnen in komplexen Dokumenten die verfügbaren Mathematik-Font-Gruppen schnell ausgehen konnten. Einziger Ausweg war das Definieren Ihrer eigenen Mathematik-Version, was ein komplizierter und mühsamer Prozess ist.

Diese Situation wurde nun durch das Schaffen eines neuen Zählers namens localmathalphabets verbessert. Dieser steuert, wie viele der Mathematik-Gruppen-Slots lokal zugewiesen werden, wenn ein neues Mathematik-Alphabet (und eine neue Mathematik-Gruppe) erstellt wird. Ist die aktuelle Formel abgeschlossen, wird jede solche weitere (lokale) Allokation zurückgenommen, so dass Sie eine reelle Chance haben, in der nächsten Formel andere neue Mathematik-Alphabete einzusetzen.

Der Standardwert für localmathalphabets ist 2, aber wenn Sie aufgrund der Komplexität Ihres Dokuments mehr lokale Alphabete benötigen, können Sie den Zähler auf einen höheren Wert wie 4 oder 5 setzen. Es ist auch möglich, ihn noch höher zu setzen, aber das wird selten hilfreich sein, weil viele Gruppen-Slots von Symbol-Fonts genutzt werden, die dann permanent alloziert sind – egal ob sie zum Einsatz kommen oder nicht. *(Github Issue 676)*

Neuer Standardwert für \tracinglostchars

Im Jahr 2021 wurden alle TEX-Engines so erweitert, dass \tracinglostchars den Wert 3 unterstützen, um fehlende Zeichen statt in Warnungen in Fehler münden zu lassen. Durch diese Änderung der Engine wurde uns klar, dass LaTeX einen besseren Standardwert für diesen Parameter setzen sollte (zuvor wurde die Warnung nur in die Transcript-Datei geschrieben). Der Einsatz des nun verfügbaren Werts von 3 als Standardwert wäre ideal, aber aus Gründen der Kompatibilität haben wir ihn im Kernel nur auf 2 gesetzt.

Aber wir empfehlen, `\tracinglostchars=3` entweder in einem Paket oder in der Präambel Ihres Dokuments zu setzen: Das Fehlen von Glyphen in der Ausgabe ist definitiv ein Fehler und sollte daher auch als solcher kenntlich gemacht werden (um sicherzustellen, dass ihm auch entsprechende Aufmerksamkeit gewidmet wird). Weitere Gründe für diese empfohlene Änderung, die insbesondere im Umfeld von Unicode-Engines liegen, werden weiter unten erläutert (in Verbindung mit der missbräuchlichen Verwendung von Textakzenten im Mathematik-Modus).

Hinzugefügte `\PackageNote` und `\ClassNote`

LaTeX bietet diese drei Befehle an: `\PackageError` zum Signalisieren von Fehlern, die die Verarbeitung stoppen; `\PackageWarning` zum Erzeugen einer Warnmeldung am Terminal mit Fortführen der Verarbeitung; und `\PackageInfo` zum Bereitstellen von Informationen, die nur in die `.log`-Datei geschrieben, aber nicht am Terminal ausgegeben werden. Bisher gab es keine Möglichkeit, Informationen am Terminal auszugeben, denen man ansieht, dass sie von einem spezifischen Paket kommen, die dabei aber keine Warnung sind. (Pakete, die etwas am Terminal ausgeben wollten, haben deshalb in der Vergangenheit oft `\PackageWarning` genutzt, auch wenn die Information gar keine Warnung war.)

Wir haben daher nun `\PackageNote` (und das eng damit verbundene `\PackageNoteNoLine`) ergänzt – beides wird als »informell« betrachtet, aber trotzdem am Terminal ausgegeben und nicht nur in die `.log`-Datei geschrieben. Ähnliche Befehle existieren für Klassen, daher gibt es auch dort nun neue Befehle: `\ClassNote` und `\ClassNoteNoLine`. *(Github Issue 613)*

Neuer Befehl `\ShowFloat`

Das Paket `fltrace` bietet eine (recht technische, aber sehr detaillierte) Möglichkeit, die Float-Mechanismen von LaTeX zu tracen. Das kann dabei helfen, zu verstehen, warum ein bestimmtes Float an einer bestimmten Stelle platziert wird oder warum es unerwartet auf einer späteren Seite erscheint. LaTeX legt Floats in Registern mit den Namen `\bx@A`, `\bx@B` und so weiter ab und diese Namen erscheinen in den Tracing-Informationen.

Um den Inhalt eines Float-Registers auszugeben, können Sie jetzt den Befehl `\ShowFloat{`*identifier*`}` verwenden, wobei es sich bei *identifier* um den (oder die) Großbuchstaben nach `bx@` aus den Registernamen handelt, die im Tracing angezeigt werden. Wenn zusätzliche Register alloziert wurden (mit `\extrafloats`), kann es sich bei *identifier* auch um eine Zahl handeln. Der Befehl ist allgemein verfügbar – egal, ob Sie `fltrace` geladen haben oder nicht – denn er ist auch beim Interpretieren der Tracing-Ausgabe des Pakets `fewerfloatpages` hilfreich.

Neues Argument für \counterwithin/without

Die Befehle \counterwithout und \counterwithin besitzen nun beide ein zusätzliches optionales Argument – ähnlich dem des Befehls \numberwithin aus amsmath, für den diese nun der zu bevorzugende Ersatz sind. Dieses optionale Argument legt das Format des Zählers wie zum Beispiel \roman fest – Standardwert ist \arabic. Alternativ kann man die Sternform der Befehle verwenden: in diesem Fall bleibt das Format des Zählers unverändert.

Prüfung auf geladene Pakete und Klassen

Um zu prüfen, ob ein Paket geladen wurde, können Sie nun \IfPackageLoadedTF {⟨*package*⟩} {⟨*true*⟩} {⟨*false*⟩} nutzen und basierend auf dem Ergebnis unterschiedlichen Code ausführen. Es ist auch möglich, zu testen, ob das Paket mit bestimmten Optionen geladen wurde. Das geschieht mit \IfPackageLoadedWithOptionsTF und vier Argumenten: {⟨*package*⟩}{⟨*option-list*⟩}{⟨*true*⟩}{⟨*false*⟩}. Dabei kommt der ⟨*false*⟩-Code zum Einsatz, wenn eine oder mehrere der folgenden Optionen in der ⟨*option-list*⟩ beim Laden des Pakets nicht spezifiziert wurden oder wenn das Paket nie geladen wurde. Beide Befehle können überall im Dokument verwendet werden, sie sind nicht auf die Präambel beschränkt.[4]

Für Klassen stehen ähnliche Befehle bereit, wobei Package im Namen durch Class zu ersetzen ist. *(Github Issue 621)*

Besserer Umgang mit einer missbräuchlichen Verwendung von \include

Der Befehl \include wurde in der Vergangenheit recht häufig inkorrekt verwendet, um eine Vielzahl von Dateien in der Präambel des Dokuments (vor \begin{document}) zu laden, was unerwünschte Nebeneffekte hatte (auch wenn diese nicht zu einer expliziten Fehlermeldung führten). Daher warnt LaTeX nun vor dieser fehlerhaften Verwendung von \include, lädt dann aber weiterhin die angegebene Datei, sofern diese vorhanden ist. Allerdings werden jetzt .aux-Datei-Einstellungen nicht mehr verändert und auch die \include-Datei-Hooks werden nicht mehr ausgeführt (das heißt lediglich die generischen Datei-Hooks aus \InputIfFileExists werden ausgeführt). *(Github Issue 645)*

[4] Das gilt jetzt auch für die korrespondierenden internen Befehle, zum Beispiel \@ifpackageloaded, die diese Beschränkung in der Vergangenheit hatten.

Verbesserungen am Code

OpenType-Versionen des Fonts Latin Modern Upright Italic verwenden

Wird ein Latin-Modern-Font mit der TU-Kodierung unter XeTeX oder LuaTeX und der Form `ui` angefordert, nutzt LaTeX nun die OpenType-Version des Fonts, statt die (T1-kodierte) Type-1-Version zu verwenden.

Zusätzliche Extended-Latin-Zeichen vordefiniert

Mehr Zeichen, wie zum Beispiel ḱ (U+1E131), sind nun vordefiniert und erfordern keine Deklaration mehr per `\DeclareUnicodeCharacter`. *(Github Issue 593)*

`\endfoo` in `\NewDocumentEnvironment` prüfen

Der Befehl `\newenvironment` hat immer geprüft, dass weder `\foo` noch `\endfoo` existieren, bevor eine Umgebung `foo` erstellt wurde. Im Gegensatz dazu hat der kürzlich eingeführte Befehl `\NewDocumentEnvironment` nur auf `\foo` geschaut und damit ein eventuell existierendes `\endfoo` überschrieben. Das wurde korrigiert und das Verhalten von `\NewDocumentEnvironment` entspricht nun dem von `\newenvironment` – lediglich die ausgegebenen Fehlermeldungen sind leicht unterschiedlich formuliert.

Die Fehlermeldung `\begin ended by …` verbessern

In der Vergangenheit konnte es passieren, dass man eine Fehlermeldung wie »`\begin{foo} ended by \end{foo}`« erhielt. Das war möglich, wenn der Umgebungsname teilweise in einem Makro verborgen war. In diesem Fall sah der Test, der die Argumente von `\begin` und `\end` verglich, möglicherweise unterschiedliche Zeichenfolgen, die dann aber in der Fehlermeldung vollständig expandiert wurden und deshalb gleich aussahen. Das hat sich nun geändert, so dass die Fehlermeldung nachvollziehbarer wird und jetzt etwa als »`\begin{foo} ended by \end{\bar}`« erscheint. *(Github Issue 587)*

Alle Argumente von `\contentsline` berücksichtigen

Ein `\contentsline`-Befehl in der `.toc`-Datei enthält immer vier Argumente, wobei das letzte normalerweise leer ist – außer beim Einsatz des `hyperref` Pakets. Die normale Definition des Befehls `\contentsline` benutzte nur die ersten drei Argumente und verließ sich darauf, dass das letzte leer ist (und somit keinen Schaden anrichtet). Aber diese Annahme ist nicht immer korrekt: wurde beispielsweise anfänglich `hyperref` geladen, dann aber zu einen späteren Zeitpunkt entfernt, enthielt das vierte Argument Daten und der nachfolgende Lauf produzierte diverse

Fehlermeldungen. Jetzt werden alle vier Argumente von `\contentsline` berücksichtigt, wobei das vierte gesichert wird, so dass es von `hyperref` genutzt werden kann. *(Github Issue 633)*

Entfernen einer Mathematik-Liste (`mlist`) in einem LuaTeX-Callback zulassen

Es führt nicht mehr zu einem Fehler, wenn der Callback-Handler bei den LuaTeX-Callbacks `pre_mlist_to_hlist_filter` und `post_mlist_to_hlist_filter` auf Entfernen der gesamten `mlist` hinweist. *(Github Issue 644)*

Erweitertes Label-Handling in Paket-Code

Seit 2020 hat LaTeX (siehe LaTeX News 32 [5]) die Namen der Zähler, die mit dem aktuellen Label verbunden sind, im internen Befehl `\@currentcounter` erfasst. Diese Möglichkeit (ursprünglich dem Paket `zref` von Heiko Oberdiek entstammend) kann genutzt werden, um Präfixe wie »Abbildung« vor dem Referenztext zu erzeugen, solange der Zähler nicht unterschiedliche Objekte in einer zusammengehörigen Folge zählt (zum Beispiel Lemmata und Theoreme). In den meisten Fällen wird das aktuelle Label durch `\refstepcounter` gesetzt, was automatisch den Namen des Zählers speichert; aber einige Konstrukte (Ausrichtungen und Fußnoten) müssen das aktuelle Label möglicherweise direkt speichern, so dass es in diesen Fällen sinnvoll ist, zusätzlich `\@currentcounter` zu aktualisieren, um den zugehörigen Zählernamen zu speichern.

In diesem Release wurden sowohl der Fußnoten-Befehl im Kernel als auch einige Umgebungen aus dem Paket `amsmath` entsprechend angepasst. Wir ermutigen die Betreuer jeglicher Klassen- oder Paket-Dateien, die `\@currentlabel` definieren, auch `\@currentcounter` an derselben Stelle zu setzen. *(Github Issue 300, 687)*

Bessere Meldung, wenn Textakzente im Mathematik-Modus verwendet werden

Der Einsatz von Textakzenten wie `\^` im Mathematik-Modus funktioniert nicht (und TeX stellt explizit Mathematik-Akzente wie `\hat` bereit, um auf solche Symbole im Mathematik-Modus zuzugreifen). Daher hat LaTeX eine Warnung ausgegeben, wenn solch ein falsch platzierter Akzent auftauchte, worauf oft ein seltsamer und scheinbar nicht damit in Verbindung stehender Low-Level-Fehler folgte. Das wurde nun geändert, so dass die Meldung zu diesem Fehler Akzente zumindest erwähnt, damit sie hoffentlich weniger verwirrend ist.

Die Diskussion über solche Warnungen oder Fehler erinnert uns daran, hier eine Empfehlung aus diesem Newsletter zu wiederholen, die im Zusammenhang mit

dem Wert von \tracinglostchars gegeben wurde. Mit TeX-Implementierungen seit 2020 können alle Warnungen, die fehlende Zeichen betreffen, in einen Fehler umgewandelt werden, indem \tracinglostchars auf 3 gesetzt wird. Wir empfehlen daher nun, diese Einstellung auf 3 vorzunehmen, insbesondere für Unicode-Engines, bei denen solche fehlenden Zeichen eher vorkommen (weil kein Font den kompletten Unicode-Bereich abdeckt). *(Github Issue 643)*

Fehlerkorrekturen

Wiederholung von Argumentprozessoren in Befehlsdeklarationen mit »embellishments«

Es gab einen Fehler in ltcmd (ehemals xparse), der dazu führte, dass sich Befehle falsch verhielten, wenn sie mit »embellishments« (e-Argument) und Argumentprozessoren definiert waren. In diesem Fall wurde nur ein (möglicherweise ungültiger) Argumentprozessor zum vollständigen Satz von e-Argumenten hinzugefügt, was in einigen Fällen zu wenig Prozessoren ergab und somit zu unvorhersehbarem Verhalten führte. Dieser Fehler wurde behoben, indem auf alle zusammengehörigen »embellishments«-Verzierungen dieselben Argumentprozessoren angewandt wurden, so dass eine Deklaration wie:

```
\NewDocumentCommand\foo{>{\TrimSpaces}e{_^}}{(#1)[#2]}
\foo^{ a }_{ b }
```

nun \TrimSpaces korrekt auf beide Argumente anwendet. *(Github Issue 639)*

Korrekte Änderung der Groß- und Kleinschreibung für \ij und \IJ

Die Ligaturen »ij« und »IJ«, die im Niederländischen zum Einsatz kommen, stehen (für die meisten TeX-Fonts) nur zur Verfügung, wenn die Befehle \ij oder \IJ verwendet werden oder wenn Sie sie als Unicode-Zeichen U+0133 beziehungsweise U+0132 eingeben. Beim Einsatz von per OT1 oder T1 kodierten Fonts in pdfTeX schlug das Umwandeln in Groß- oder Kleinschreibung per \MakeUppercase oder \MakeLowercase allerdings unabhängig von der Eingabemethode immer fehl. Das wurde nun korrigiert. Gleichzeitig haben wir das Trennen von Wörtern mit dieser Ligatur (bei Verwenden der OT1-Kodierung) verbessert. *(Github Issue 658)*

Legacy-Änderungen von Schriftfamilien-Standards

In der Vergangenheit hat man Änderungen an den Defaults von Schriftfamilien durch direktes Anpassen von \bfdefault oder \mddefault vorgenommen. Seit 2020 gibt es nun \DeclareFontSeriesDefault, mit dem eine granulare Steuerung möglich

ist: Durch diese Deklaration können Sie den Default für einzelne Meta-Schriftfamilien anpassen, indem Sie beispielsweise die Einstellungen für fette Schriften nur für die serifenlose Schriftfamilie ändern, ohne die für `\rmfamily` oder `\ttfamily` zu beeinflussen. In [4] finden Sie weitere Details.

Aus Gründen der Abwärtskompatibilität ist es weiterhin möglich, `\bfdefault` mit `\renewcommand` zu ändern; wird dies verwendet, so wird die Einstellung für alle Meta-Familien in einem Zug geändert. Diese Änderung kann nicht vorgenommen werden, wenn der Befehl `\renewcommand` ausgeführt wird, und wurde daher bis zur nächsten Ausführung von `\bfseries` oder `\mdseries` verzögert. Das Problem bei diesem Vorgehen war allerdings, dass jeder zwischenzeitliche Aufruf von `\DeclareFontSeriesDefault` überschrieben wurde – diese beiden Vorgehensweisen passten also nicht gut zusammen. Das Problem bestand darin, dass ältere Font-Pakete die Legacy-Methode verwenden, während neuere `\DeclareFontSeriesDefault` verwenden.

Dies wurde nun gelöst, indem `\DeclareFontSeriesDefault` vor dem Setzen der neuen Standardwerte die notwendigen Resets ausführt. *(Github Issue 663)*

Verwenden von `#` in `\textbf` und ähnlichen Befehlen

Bisher konnten Sie das Makro-Parameterzeichen `#` in Inline-Funktionen innerhalb des Arguments von `\textbf` oder ähnlichen Text-Font-Befehlen nicht verwenden. Ein interner Befehl wird nun mit `\unexpanded` geschützt, so dass der Einsatz von `#` nicht mehr länger zu einem Fehler führt. *(Github Issue 665)*

Änderungen an Paketen der Kategorie `amsmath`

Verbesserte Kompatibilität mit `hyperref`

Diese Änderung in `amsmath` behebt ein Abstandsproblem, das von der in `hyperref` verwendeten Methode zum Verändern der `equation`-Umgebung verursacht wird. Der Einfachheit halber wurde `amsmath` ein expliziter, einfacher (und daher möglicherweise temporärer) Patch hinzugefügt: Dieser besteht aus einem zusätzlichen, leeren (und daher unsichtbaren) `\mathopen`-Atom (ohne mathematische Bedeutung) am Anfang des mathematischen Inhalts der Umgebung. *(Github Issue 652)*

Änderungen an Paketen der Kategorie `graphics`

`graphicx`: Neuer Schlüssel für alternativen Text

Es wurde ein neuer Schlüssel `alt` zu `\includegraphics` hinzugefügt, um beschreibenden Text hinzufügen zu können, der für die Barrierefreiheit wichtig ist. Dieser

Schlüssel wird standardmäßig nicht verwendet – er kann durch Erweiterungspakete ausgegeben werden und wird für andere zukünftige Funktionalitäten nützlich sein.
(Github Issue 651)

Änderungen an Paketen der Kategorie `tools`

`array`: Kein `\mathsurround` um ein `tabular`

Eine `tabular`-Umgebung wird (intern) als `array`-Umgebung mit speziellen Einstellungen gesetzt und verwendet daher im Hintergrund den mathematischen Modus. Da es aber sich in Wirklichkeit nicht um eine mathematische Formel handelt, sollte kein zusätzlicher Leerraum durch `\mathsurround` hinzugefügt werden (der Abstand um `tabular` sollte davon nicht beeinflusst werden). Beachten Sie, dass es diesen Fehler schon »immer« gibt, was zeigt, dass `\mathsurround` entweder nie zum Einsatz kommt – oder sein Einfluss in diesem Fall nie bemerkt wurde. Auf jeden Fall wurde dieser Fehler nun endlich behoben. *(Github Issue 614)*

`longtable`: Verbesserungen nach einer Abschnittsüberschrift

Die `longtable`-Umgebung setzt jetzt das Flag `\@nobreakfalse`, um den Schriftsatz zu korrigieren, wenn eine Tabelle unmittelbar auf eine Überschrift folgt. Zuvor wurden die Änderungen des Abstands und der Einrückung, die unmittelbar nach einer Abschnittsüberschrift erforderlich sind, fälschlicherweise innerhalb des nächsten Absatzes (falls vorhanden) nach der Tabelle ausgeführt. Zudem wurde auch eine ähnliche Prüfung für `\if@noskipsec` ergänzt, so dass eine Tabelle nach einer Run-In-Überschrift korrekt platziert wird, statt vor dieser Überschrift zu erscheinen.
(Github Issues 131 und 173)

`multicol`: Bessere Steuerung des Spaltenumbruchs

Ab Version 1.9 akzeptiert `\columnbreak` ein optionales Argument (wie `\pagebreak`), mit dem Sie angeben können, wie wünschenswert es ist, die Spalte nach der aktuellen Zeile umzubrechen: Unterstützte Werte sind `0` bis `4`, wobei höhere Zahlen eine höhere Attraktivität anzeigen. Diese Version fügt auch `\newcolumn` hinzu, das einen Umbruch erzwingt, aber die Spalte kurz hält (vergleichbar mit `\newpage` für Seiten).
(Github Issue 682)

`varioref`: Verbesserter Umgang mit fehlenden Labeln

Wird ein nicht definiertes Label referenziert, erstellt `varioref` eine Standarddefinition, so dass spätere Verarbeitungsschritte die richtige Struktur vorfinden (zwei ge-

schweifte Klammern innerhalb von \r@⟨*label*⟩). Werden aber nameref oder hyperref geladen, ändert sich diese Datenstruktur und sie enthält fünf Argumente – das kann in manchen Fällen zu Low-Level-Fehlern führen. Der Code wurde daher nun geändert, um diese Fehler zu vermeiden. *(https://tex.stackexchange.com/603948)*

Literatur

[1] LaTeX Dokumentation auf der LaTeX Project Website, https://latex-project.org/help/documentation/.

[2] Frank Mittelbach, Chris Rowley: LaTeX Tagged PDF—A blueprint for a large project, https://latex-project.org/publications/indexbyyear/2020/.

[3] »LaTeX News 34«, *42*.3 (132 2021), 305–310, ISSN: 0896-3207, https://tug.org/TUGboat/tb42-3/tb132ltnews34.pdf.

[4] LaTeX Project Team: LaTeX 2_ε News 31, https://latex-project.org/news/latex2e-news/ltnews31.pdf.

[5] — LaTeX 2_ε News 32, https://latex-project.org/news/latex2e-news/ltnews32.pdf.

[6] — LaTeX 2_ε News 33, https://latex-project.org/news/latex2e-news/ltnews33.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.

texlogfilter von *Julien Labbé* ist ist ein Perl-Skript, das Logfiles aller LaTeX-Engines filtert, so dass nur noch Warnungen und Fehlermeldungen koloriert verbleiben.
CTAN:support/texlogfilter

bmstu von *Mikael Novikov* ist eine Klasse für Arbeiten an der Staatlichen Technischen Universität Moskau.
CTAN:macros/latex/contrib/bmstu

altsubsup von *Julien Labbé* modifiziert das Hoch- und Tieferstellen im Mathematikmodus, um den Text leichter formatieren zu können.
CTAN:macros/latex/contrib/altsubsup

`citation-style-language` von *Zeping Lee* implementiert eine Alternative zu BibTeX und Biblatex/Biber zur Literaturverwaltung und ermöglicht es, die *Citation Style Language* (CSL) mit LaTeX zu verwenden. Das Paket enthält ein Lua-Skript und eine elementare Sammlung von Zitationsstilen und Lokalisierungen in CSL sowie einen LaTeX-Stil. CSL ist ein auf XML beruhender Standard, der seit fast 20 Jahren entwickelt und in vielen Literaturverwaltungen verwendet wird. Es gibt derzeit etwa 1500 Zitationsstile im GitHub-Repositorium des CSL-Projekts.
`CTAN:biblio/citation-style-language`

`hamnosys` von *Thomas Hanke* enthält einen Unicode-Font und die entsprechende LaTeX-Unterstützung, um Zeichen nach dem Hamburger Notationssystem für Gebärdensprachen (HamNoSys) mit XeLaTeX und LuaLaTeX zu setzen.
`CTAN:fonts/hamnosys`

`kanbun` von *Yuanhao Chen* ist ein weiteres Paket zum Setzen der chinesischen Han-Schrift.
`CTAN:macros/latex/contrib/kanbun`

`llncs` von *Markus Richter* ist der CTAN-Release der Klasse und des Bibliografie-Stils zu der Reihe *Lecture Notes in Computer Science (LNCS)* aus dem Springer-Verlag.
`CTAN:macros/latex/contrib/llncs`

`concmath-otf` von *Daniel Flipo* enthält eine OpenType-Version der Schrift Concrete Math, die *Ulrik Vieth* in METAFONT erstellt hat, samt der zugehörigen LaTeX-Unterstützung.
`CTAN:fonts/concmath-otf`

`coop-writing` von *Geraldo Xexéo* unterstützt kollaboratives Schreiben mit LaTeX (Kommentare, Überarbeitungshinweise, Vorschläge, anonymer Review, Entwurf usw.).
`CTAN:macros/latex/contrib/coop-writing`

`latex-lab-dev` vom *LaTeX Team* dient zum Testen von Ergänzungen des LaTeX-Kernels. Es tritt somit neben das Paket `latex-base-dev`, das denselben Zweck für die Kernelentwicklung selbst erfüllt. Näheres dazu soll in den LaTeX News Nr. 35 nachzulesen sein, die demnächst erscheinen.
`CTAN:macros/latex-dev/required/latex-lab`

`sillypage` von *Paulo Roberto Massa Cereda* ist ein Spaß-Paket, das die Seitenzahlen durch stilisierte Figuren ersetzt, die den »Silly Walk« von *John Cleese* (*Monty Python*) zeigen.
`CTAN:macros/latex/contrib/sillypage`

`pgf-interference` von *Keno Wehr* zeichnet Interferenzmuster, die entstehen, wenn monochromatisches Licht an regelmäßigen Schlitzen gebeugt wird.
`CTAN:graphics/pgf/contrib/pgf-interference`

biblatex-readbbl von *Herbert Voß* verändert das Makro von biblatex, das die bbl-Datei einliest. Dadurch kann man die von Biber vorbereitete bbl-Datei mit der filecontents-Umgebung in ein Dokument einfügen und weitergeben.
CTAN:macros/latex/contrib/biblatex-contrib/biblatex-readbbl

pascaltriangle von *Nan Geng* ermöglicht das Zeichnen von Yang-Hui-Dreiecken, der ältesten Darstellung des Pascalschen Dreiecks, die auf den chinesischen Mathematiker Yang Hui aus dem 13. Jahrhundert zurückgeht.
CTAN:macros/latex/contrib/pascaltriangle

dbshow von *Changkai Li* speichert und stellt Daten mit anpassbaren Vorlagen und Filtern dar.
CTAN:macros/latex/contrib/dbshow

yb-book von *Yegor Bugayenko* ist eine Klasse, die der Autor für seine eigenen Bücher verwendet hatte, die er bei Amazon veröffentlichte.
CTAN:macros/latex/contrib/yb-book

wrapfig2 von *Claudio Beccari* ist eine Neufassung des Pakets wrapfig von *Donald Arseneau* in LaTeX3, mit dem man Text um Formen herum fließen lassen und zusätzlich farblich hinterlegt ausgeben kann.
CTAN:macros/latex/contrib/wrapfig2

mathalphabets von *Conden Chao* ist eine kurze chinesische Einführung in den Einsatz von griechischen und lateinischen Buchstaben beim Mathematiksatz.
CTAN:info/mathalphabets

codebox von *Nan Geng* setzt Quelltext-Listings in eine Box mit Rahmen.
CTAN:macros/latex/contrib/codebox

kaytannollista-latexia von *Teemu Likonen* ist eine fast 300-seitige Einführung zu LaTeX auf Finnisch.
CTAN:info/kaytannollista-latexia

bfh-ci von *Marei Peischl* ist das Corporate Design für Drucksachen und Präsentationen, die an der Berner Fachhochschule erstellt werden.
CTAN:macros/latex/contrib/bfh-ci

kinematikz von *Vitor Santos* dient zum Zeichnen von kinematischen Ketten mit Hilfe von pgf/Ti*k*Z.
CTAN:graphics/pgf/contrib/kinematikz

termsim von *Nan Geng* simuliert die Terminals von Windows 10, Ubuntu und macOS, jeweils mit dunklem, hellem und weißem Farbschema bei der Fensterdarstellung.
CTAN:macros/latex/contrib/termsim

autopuncitems von *Kale Ewasiuk* erweitert die Aufzählungsumgebungen in enumitem um eine automatisierte Zeichensetzung am Ende jeder Aufzählungs-

punkts.
CTAN:macros/luatex/latex/autopuncitems

pst-hsb von *Herbert Voß* ist ein PSTricks-Paket, mit dem man Funktionsgraphen plotten und mit Farbverlauf einfärben kann.
CTAN:graphics/pstricks/contrib/pst-hsb

texlogsieve von *Nelson Lago* ist ein texlua-Skript, das ein LaTeX-Logfile einliest und zu einem zusammenfassenden Bericht verarbeitet.
CTAN:support/texlogsieve

numerica-tables von *Andrew Parsloe* ist aus dem Paket numerica desselben Autors ausgegliedert und dient zum Erstellen mehrspaltiger Tabellen von Funktionswerten.
CTAN:macros/latex/contrib/numerica-tables

numerica-plus von *Andrew Parsloe* ist ebenfalls aus dem Paket numerica desselben Autors ausgegliedert und dient zur Kurvendiskussion.
CTAN:macros/latex/contrib/numerica-plus

unbtex von *Henrique Cezar Ferreira* ist eine Klasse für wissenschaftliche Arbeiten an der Universität Brasília.
CTAN:macros/latex/contrib/unbtex

rbt-mathnotes von *Rebecca Turner* enthält einige Makros, mit denen die Autorin ihre Hausarbeiten und »Formel-Spickzettel« gesetzt hatte.
CTAN:macros/latex/contrib/rbt-mathnotes

njuvisual von *Yu Xiong* enthält die Logos der Universität Nanjing als Ti*k*Z-Zeichnungen.
CTAN:macros/latex/contrib/njuvisual

hep-float von *Jan Hajer* ändert die Standard-Definitionen für Gleitobjekte und lädt zusätzliche Pakete, die hilfreich sind, um die Platzierung von Gleitobjekten zu erleichtern.
CTAN:macros/latex/contrib/hep-float

hep-text von *Jan Hajer* erweitert das Paket enumitem von *Javier Bezos* und führt eine Handvoll kleinere Makros ein, die für den Satz von Texten hilfreich sein können.
CTAN:macros/latex/contrib/hep-text

hep-math von *Jan Hajer* erweitert die Pakete mathtools und amsmath.
CTAN:macros/latex/contrib/hep-math

hep-bibliography von *Jan Hajer* erweitert das Paket biblatex um alle BibTeX-Felder, die *Discover High-Energy Physics Content* unter https://inspirehep.net/ bereitstellt.
CTAN:macros/latex/contrib/biblatex-contrib/hep-bibliography

hep-acronym von *Jan Hajer* stellt ein Akronym-Makro bereit, das auf dem Paket glossaries von *Nicola Talbot* beruht.
CTAN:macros/latex/contrib/hep-acronym

hep-title von *Jan Hajer* erweitert die Titelseite der Standardklassen um weitere Angaben: Preprint, Zugehörigkeit zu einer Institution, Herausgeber und Unterstützer der Veröffentlichung.
CTAN:macros/latex/contrib/hep-title

hep-math-font von *Jan Hajer* passt die Schrift im Mathematikmodus automatisch an, wenn das ganze Dokument in einer serifenlosen Schrift gesetzt wird. Griechische Buchstaben werden im Mathematikmodus kursiv und sonst aufrecht gesetzt.
CTAN:fonts/utilities/hep-math-font

hep-font von *Jan Hajer* lädt die Schriftart Latin Modern und ergänzt fehlende Fonts aus Computer Modern.
CTAN:fonts/utilities/hep-font

snaptodo von *Hsin-Po Wang* versteht sich als eine Alternative zu dem Paket todonotes von *Henrik Skov Midtiby*. Es unterscheidet sich davon minimal: Die Randnotizen werden in denjenigen Rand gesetzt, der beim Aufruf des Makros näher liegt; sie überlappen sich nicht, und man kann ihre Formatierung steuern.
CTAN:macros/latex/contrib/snaptodo

luafindfont von *Herbert Voß* ist ein Skript, das in der LuaTeX-Font-Datenbank nach lokal installierten Schriften sucht, die mit LuaLaTeX und XeLaTeX verwendet werden können. Die Ausgabe ist umfassender als bei luaotfload-tool, weil auch trunkierte Suchen durchgeführt werden. Das Skript benötigt Lua 5.3.
CTAN:support/luafindfont

zref-clever von *Gustavo Barros* erweitert das Paket zref um Referenztypen und -formate, die Querverweise vereinheitlichen und vereinfachen.
CTAN:macros/latex/contrib/zref-clever

formal-grammar von *Martin Vassor* ermöglicht das Setzen formaler (kontextfreier) Grammatiken in Backus-Naur-Form (BNF).
CTAN:macros/latex/contrib/formal-grammar

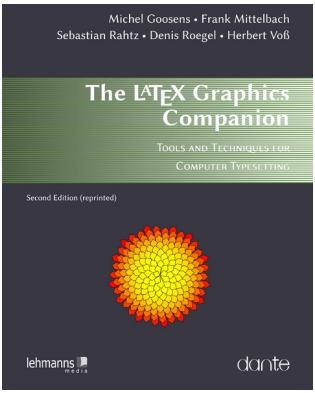
bodeplot von *Rushikesh Kamalapurkar* dient zum Zeichnen von Bode-, Nyquist- and Nichols-Diagrammen mit Hilfe von gnuplot oder pgfplots.
CTAN:graphics/pgf/contrib/bodeplot

ccref von *Jinwen Xu* erweitert das Paket cleveref, so dass bei Texten in romanischen Sprachen auch der notwendige Partitivartikel automatisch eingefügt wird, damit grammatikalisch korrekte Ausdrücke entstehen.
CTAN:macros/latex/contrib/ccref

Bücher

Edition dante – Reprint

M. Goosens, F. Mittelbach, S. Rahtz,
D. Roegel, H. Voß:
The LaTeX Graphics Companion;
2. Auflage 2022 (reprint),
DANTE e.V. und Lehmanns Media,
975 Seiten; ISBN 978-3-96543-303-8;
39,95 € (Ladenpreis) bzw. 35,– € für Mitglieder
von DANTE e.V., jeweils versandkostenfrei.



Bestellung

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

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

Spielplan

15. 3. 2022 **Tag der Druckkunst**
Bundesverband bildender Künstlerinnen und Künstler
https://www.tag-der-druckkunst.de/

23. 6. – 25. 6. 2022 **DANTE 2022**
und 64. Mitgliederversammlung von DANTE e.V.
Otto-von-Guericke-Universität
39016 Magdeburg
https://www.dante.de/veranstaltungen/dante2022/

22. 7. – 24. 7. 2022 **TUG 2022 – Presentations covering the TEX world**
The 43rd Annual Conference of the TEX Users Group
Die Tagung findet erneut online statt.
https://www.tug.org/tug2022/

September 2022 **Herbsttagung**
und 65. Mitgliederversammlung von DANTE e.V.
Landesinstitut für Pädagogik und Medien
Beethovenstraße 26
66125 Saarbrücken

12. 9. – 18. 9. 2022 **ConTEXt Meeting 2022**
Evangelische Freizeitstätte in Dreifelden/Westerwald
https://www.rittersmorgen.de

Stammtische

In verschiedenen Städten im Einzugsbereich von DANTE e.V. finden regelmäßig Treffen von TEX-Anwendern statt, die für jeden offen sind. Im Web gibt es aktuelle Informationen unter https://projekte.dante.de/Stammtische/WebHome.

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
pausiert z. Zt. wegen Corona

Berlin
Michael-E. Voges, Tel.: 0 33 62/ 50 18 35,
z .Zt. online über jitsi-meet https://meet.ffmuc.net/LaTeXStammtischBerlin

»Mantée« – Café Restaurant, Chausseestr. 131, 10115 Berlin
Zweiter Donnerstag im Monat, 19:00 Uhr

Darmstadt
Karlheinz Geyer
geyerk@posteo.de
pausiert z. Zt. wegen Corona
findet vorrausichtlich 2022 an jedem ersten Freitag im Monat wieder statt.

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
z. Zt. inaktiv. Bei Fragen steht Günther gern per Mail zur Verfügung.

Hannover

Reiko Kaps
kaps@luis.uni-hannover.de
http://tex-hannover.de/
Zweiter Donnerstag im Monat, 18:30 Uhr
z. Zt. online über https://vc.sonia.de/TexStammtischHannover

Heidelberg

Martin Wilhelm Leidig, Tel.: 01 70 41 83 32 9,
moss@moss.in-berlin.de
Anmeldeseite zur Mailingliste: https://tinyurl.com/stammtisch-HD
Physische Treffen bleiben ausgesetzt, solange keine dauerhafte Besserung bzw. Beendigung der gegenwärtigen pandemischen Lage eingetreten ist.

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

Leipzig
Erhard Pross
Erhard.Pross@gmx.de
Ab 2022 geplant. Interessenten bitte per Mail melden

München
Uwe Siart
uwe.siart@tum.de,
http://www.siart.de/typografie/stammtisch.xhtml
pausiert z. Zt. noch wegen Corona

Stralsund
Heiner Richter
Heiner.Richter@hochschule-stralsund.de
z. Zt. inaktiv, Nachfolge in Planung

Stuttgart
Bernd Raichle
bernd.raichle@gmx.de
Zweiter Dienstag im Monat, 19:30 Uhr
»Paulaner am Alten Postplatz«, Calwer Str. 45, 70173 Stuttgart

Wuppertal
Andreas Schrell
as@schrell.de
Zweiter Donnerstag im Monat, 19:30 Uhr
»Restaurant Croatia«, Südstr. 10, 42103 Wuppertal

Adressen

DANTE, Deutschsprachige Anwendervereinigung TEX e.V.
Postfach 11 03 61
69072 Heidelberg

Tel.: (0 62 21) 2 97 66 (Mo., Mi., Do. von 9.00–11.30 Uhr)
Fax: (0 62 21) 16 79 06
E-Mail: info@dante.de

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

Vorstand

Vorsitzender:	Martin Sievers	president@dante.de
stv. Vorsitzender:	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	
	Herbert Voß	

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

TEXnische Fragen

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

Autoren/Organisatoren

Doris Behrendt [9]
siehe Seite 98

Adelheid Bonnetsmüller [18]
bonnetsmueller@icloud.com

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

TEX-Stammtisch Erlangen [10]

Jürgen Fenn [88]
Neu-Isenburg
juergen.fenn@gmx.de

Rainer-M. Fritsch [38]
Speerweg 67
13465 Berlin
mail@rmf.berlin

Tobias Hilbricht [32]
hilbricht@linopus.de

Mathias Magdowski [6]
Otto-von-Guericke-Universität
Magdeburg
mathias.magdowski@ovgu.de

Ralf Mispelhorn [29]
Am Wettbach 12/2
72336 Balingen
mispelsoft@mispelhorn.de

Frank Mittelbach [75]
LATEX Project
frank.mittelbach@latex-project.org

Rolf Niepraschk [12]
Persiusstr. 12
10245 Berlin
Rolf.Niepraschk@gmx.de

Henning Hraban Ramm [57]

Thomas A. Schmitz [60]

Martin Sievers [4,6]
siehe Seite 98

Herbert Voß [49,93]
Wasgenstraße 21
14129 Berlin
herbert@dante.de

Die TEXnische Komödie

34. Jahrgang Heft 1/2022 Februar 2022