Die TEXnische Komödie

dante

Deutschsprachige
Anwendervereinigung TEX e.V.

20. Jahrgang Heft 3/2008 August 2008

3/2008

Impressum

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

Diese Ausgabe wurde mit `pdfTeX 3.1415926-1.40.8-2.2 (Web2C 7.5.7)` erstellt. Als Standard-Schriften kamen die Type-1-Fonts Latin-Modern und LuxiMono zum Einsatz.

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

Redaktionsschluss für Heft 4/2008: 15. Oktober 2008 ISSN 1434-5897

Editorial

Liebe Leserinnen und Leser,

im letzten Heft gab es den Tagungsbericht zu DANTE 2008, während Sie in dieser Ausgabe den sogenannten »Call for Papers« für die Herbsttagung in Tübingen finden. Dieses Editorial entsteht, während gerade die TUG-Tagung in Cork stattfindet. Die Aktivitäten der TEX-Gemeinde sind immer noch sehr vielfältig und lassen auch nicht in ihrem Eifer nach. Auch dieses Mal enthält unsere Rubrik »Neue Pakete« wieder eine sehr lange Liste. Weiterhin finden Sie einige Fotos vom Linuxtag 2008 in Berlin.

Ulrike Fischer gibt eine Einführung in die relativ neue TEX-Entwicklung XETEX, welche anfänglich nur für Mac OS zugänglich war. Jonathan Kew, der Autor von XETEX, hat nun eine funktionsfähige Version für Linux und Christian Schenk eine für Windows erstellt. Der Umgang mit dieser TEX-Version erfordert einige grundlegende Kenntnisse, die Ihnen der Artikel zugänglich macht. Uwe Ziegenhagen erläutert in seinem Beitrag die Arbeit mit Subversion, einem Revisions-Kontrollsystem.

Fehler gehören zum Alltag, was man wieder einmal an der letzten Ausgabe feststellen konnte. Eine einzige Zeile für die Definition der `lstlisting`-Umgebung des Artikels zu `biblatex` gelangte nicht in die Hauptdatei, sodass bedauerlicher Weise einige Listings des Artikels fehlerhaft waren. Es dürfte wohl nicht zu größeren Problemen beim Lesen gekommen sein, da dieser Fehler nicht unbedingt das Verständnis des Artikels behindert hat. Der zweite Teil zu `biblatex` wird dann hoffentlich fehlerfrei in einer der nächsten Ausgaben erscheinen.

Ich wünsche Ihnen viel Spaß bei der Lektüre und verbleibe
mit TEXnischen Grüßen,

Ihr Herbert Voß

Hinter der Bühne

Vereinsinternes

Grußwort

Liebe Mitglieder,

viele Grüße von der TUG 2008 in Cork. Irland präsentiert sich uns sehr freundlich und an einigen Stellen anders als manchen Vorurteilen folgend: Es ist sonnig und trocken, das Mittagessen in der Kantine ist schmackhaft, und das reichhaltige irische Frühstück besteht im Studentenwohnheim aus zwei Scheiben pappigem Brot, einem Stück Butter, einem kleinen Töpfchen Marmelade und Instant-Kaffee.

Diese Ausgabe von »Die TEXnische Komödie« enthält die Einladung zur Herbsttagung in Tübingen. Einer der Tagesordnungspunkte sieht das Einstellen neuer Mittel in den Projektfonds von DANTE e.V. vor. Der Abfluss der Mittel an Projekte zur Weiterentwicklung von TEX und Co. entwickelte sich seit dem letzten Mittelzufluss aus Vereinsmitteln vor zwei Jahren weitgehend nach Plan. Zwischenzeitlich gab es eine Spende der Firma QuinScape aus Dortmund an den Projektfonds, für die wir uns an dieser Stelle noch einmal bedanken möchten. Ein weiterer Tagesordnungspunkt sieht einen optionalen Versand von Beitragsrechnung und Zuwendungsbescheinigung per E-Mail vor. Da die Deutsche Post AG uns mitgeteilt hat, dass sie den bisher durchgeführten Versand dieser Schreiben als Beilage zu »Die TEXnische Komödie« nach den Bedingungen der Presse-Distribution als unzulässig ansieht, müssen wir diese zukünftig separat als Brief verschicken. Um Porto zu sparen, planen wir daher, die Rechnung und Zuwendungsbescheinigung denjenigen Mitgliedern, die sich damit einverstanden erklären, per E-Mail als PDF-Datei zuzusenden. Wir freuen uns auf alle Mitglieder, die wir in Tübingen begrüßen können.

Mit freundlichem Gruß,

Klaus Höppner	Volker RW Schaa
Vorsitzender	Stellvertretender Vorsitzender

Einladung und »Call for Papers« zur Herbsttagung von DANTE e.V.

Klaus Höppner, Wolfgang Engelmann

Liebe Mitglieder von DANTE,

die nächste TEX-Tagung von DANTE e.V. findet am Samstag, den 13. September 2008 an der Universität Tübingen statt.

Programmpunkte sind kostenlose Tutorien und die 39. Mitgliederversammlung.

Die Tagesordnung der Mitgliederversammlung um 9.00 Uhr an der

Eberhard-Karls-Universität Tübingen
Wilhelm-Schickard-Institut für Informatik
Raum A301
Sand 13
72076 Tübingen

lautet:

1. Begrüßung und Tagesordnung
2. Bericht des Vorstands
3. Bereitstellung neuer Mittel für den Projektfonds
4. Option zum Versand der Beitragsrechung und Zuwendungsbescheinigung per E-Mail
5. Verschiedenes

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 willkommen.

Falls Sie ein Tutorium oder einen Vortrag anbieten wollen, werden Sie gebeten, dies mit dem Anmeldeformular unter `http://www.dante.de/dante/events/mv39/` oder per E-Mail an `mv39@dante.de` bei den Organisatoren bis 30. August 2008 anzumelden.

Zu einem Vortrag oder Tutorium ist ein Abstract als Text- oder LaTeX-Datei einzureichen. Dieser soll maximal eine Seite umfassen.

Firmen und Institutionen, die ihre Produkte präsentieren bzw. die Tagung finanziell unterstützen wollen, werden gebeten, sich frühzeitig an dieselben Adressen zu wenden.

Die Homepage der Tagung findet sich unter `http://www.dante.de/dante/events/mv39/`.

Mit Fragen, Wünschen und Anregungen wenden Sie sich bitte an

DANTE e.V.
Stichwort: Mitgliederversammlung von DANTE e.V.
Postfach 10 18 40
69008 Heidelberg
E-Mail: `mv39@dante.de`

Mit freundlichen Grüßen,

Klaus Höppner (DANTE e.V.)
Wolfgang Engelmann (Tübingen)

Bretter, die die Welt bedeuten

Erste Schritte mit XeLaTeX

Ulrike Fischer

Der folgende Artikel gibt eine kleine – wenigstens fing sie klein an ... – Einführung in die Verwendung von XeTeX mit dem LaTeX-Format («XeLaTeX»).

XeTeX bietet

- echte Unicode-Unterstützung bei der Ein- und Ausgabe,
- die einfache Benutzung von Schriften, die im System installiert sind. Eine gesonderte Installation der Schriften für TeX ist nicht mehr nötig.
- Unterstützung von OpenType-Schriften.

XeTeX wurde vor ca. vier Jahren ursprünglich für den Mac entwickelt. Der Autor heißt Jonathan Kew. Seine Homepage ist `http://scripts.sil.org/XeTeX`. Dort gibt es auch ein Repository [14] (im Folgenden XeTeX-SVN genannt) mit den neuesten Versionen der Quelldateien und diverser XeTeX-spezifischen Pakete. Bugreports kann man auf der Sourceforge-Seite `https://sourceforge.net/projects/xetex/` machen. In MiKTeX gibt es XeTeX seit der Version 2.7, also offiziell seit Dezember 2007.

XeTeX ist immer noch im Betastadium – die aktuelle Versionsnummer ist 0.999.0 –, man sollte also immer noch mit Bugs und mit inkompatiblen Änderungen rechnen. Beispielsweise wurde in Version 0.996, die es für Windows nie gegeben hat, als Default-pdf-Treiber noch `xdv2pdf` benutzt, seit Version 0.997 ist es das deutlich leistungsfähigere `xdvipdfmx`. MiKTeX ist sehr schnell im Aktualisieren von XeTeX, gelegentlich lagen zwischen einer Diskussion über einen Bug in der Mailingliste und dem MiKTeX-Update nur wenige Tage. In anderen Systemen sind die XeTeX-Versionen häufig deutlich älter.

Die Dokumentation zu XeTeX ist (noch) etwas dünn. In MiKTeX erhält man mit `mthelp xetex` die Seite zu XeTeX aus dem MiKTeX-Manual und mit `mthelp xetexref` eine relativ kurze Referenz. Einiges zusätzliches Material zum LaTeX Graphics Companion wurde ins Netz gestellt [7]. Dort gibt es auch eine gute Dokumentation zu XeTeX von Michel Goossens [3]. Es gibt eine Mailingliste für XeTeX [13]. Es ist möglich, über den Newsserver `news.gmane.org` und die Newsgruppe `gmane.comp.tex.xetex` die Liste zu lesen und Nachrichten an sie zu schicken.

Ich beschränke mich im Folgenden auf XeLaTeX – von ConTeXt verstehe ich nichts. Ich werde auch nichts über fremde Schriften wie Chinesisch oder Arabisch sagen – davon verstehe ich auch nichts. Meine Arbeitsumgebung ist MiKTeX 2.7 und WinEdt 5.5 auf WinXP – bei anderen Betriebssystemen können Aufrufoptionen, Suchpfade, Versionen und Konfigurationen anders sein. Die grundsätzlichen Aussagen treffen aber auch auf die Verwendung von XeLaTeX unter anderen Betriebssystemen zu.

Ein Beispieldokument

Dieses Dokument kann als Startpunkt für Experimente mit XeTeX dienen:

```
\documentclass{scrartcl}
\usepackage[ngerman]{babel}
% Alternative zu babel: polyglossia.

% Darauf verzichte ich nie:
\usepackage[babel]{csquotes}
\MakeAutoQuote{«}{»}

% Entweder
\usepackage{xltxtra}

% oder zumindest:
%\usepackage{ifxetex}
%\usepackage{fontspec}
%\usepackage{xunicode}

\begin{document}
«Hallo Welt»
\end{document}
```

XeLaTeX starten

Für die ersten Tests kann man XeLaTeX von der Eingabeaufforderung aus starten.

```
xelatex --no-pdf foo
```

erzeugt aus der LaTeX-Datei `foo.tex` die Datei `foo.xdv`. Eine `xdv`-Datei ist eine Art erweiterte (*extended*) `dvi`-Datei.

```
xdvipdfmx foo
```

erzeugt dann aus der `xdv`-Datei die `pdf`-Datei `foo.pdf`.

Mit

```
xelatex foo
```

kann man beide Schritte zusammenfassen. Die `xdv`-Datei wird dabei anschließend gelöscht.

Das Endprodukt von Xe(La)TeX sind also `pdf`-Dateien. Wie jeder Windows-Benutzer wahrscheinlich schon leidvoll erfahren hat, sperrt der Adobe Reader alle geöffneten `pdf`-Dateien. Die diversen TeX-Programme und `pdf`-Treiber können dann diese Dateien nicht neu schreiben. Leider ist die Fehlermeldung von XeLaTeX (genauer: von `xdvipdfmx`) in solchen Fällen ziemlich nichtssagend. Bei der folgenden Meldung sollte man als Erstes prüfen, ob die `pdf`-Datei geöffnet ist:

```
Error 1 (driver return code) generating output;
file foo.pdf may not be valid.
```

`xelatex` kennt neben `--no-pdf` noch weitere Optionen. Die meisten entsprechen den Optionen der anderen TeX-Compiler. Interessant ist aber noch die Option `--output-driver`. Mit ihr kann man den Treiber einstellen, der für den `xdv->pdf`-Schritt benutzt wird. Zwar gibt es in Windows keine Alternative zu `xdvipdfmx` – bei Mac-OS gibt es noch `xdv2pdf` –, aber die Option kann auch dazu dienen, Optionen an `xdvipdfmx` weiterzureichen. Bei Problemen kann man z. B. mit `xelatex --output-driver="xdvipdfmx -vv" foo` den Verbosemodus von `xdvipdfmx` einschalten.

XeLaTeX in WinEdt einbauen

Auf die Dauer ist es natürlich bequemer, XeLaTeX über den Editor aufzurufen. Zum einen spart man sich die Tipparbeit (und die Tippfehler), zum anderen kann WinEdt (TeXnicCenter meines Wissens auch) geöffnete `pdf`-Dateien vor der Kompilation schließen.

WinEdt hat derzeit noch keine vordefinierten Menüs und Befehle für XeLaTeX. Als muss man sie selbst hinzufügen. Dies wird sich mit WinEdt 5.6, das es derzeit (Mai 2008) als Beta 1 gibt, ändern.

1. Als Erstes braucht man einen Eintrag für XeLaTeX im Menü `Options-> Execution Modes`.

   Dazu muss man im WinEdt-Ordner nach der Datei `Winedt\Exec\MiKTeX\ ExecModes.edt` suchen und eine Kopie dieser Datei unter gleichem Namen und in der gleichen Ordnerstruktur in den Anwendungsdaten von WinEdt ablegen (den Ordner findet man über den Configuration Wizard und den Knopf «Browse Application Data»). Anschließend öffnet man diese Kopie in WinEdt und kopiert den zu pdfLaTeX gehörenden Abschnitt, fügt ihn da ein, wo man ihn haben will, und ändert im kopierten Text überall «`pdflatex`» in «`xelatex`».

2. Als Nächstes braucht man eine WinEdt-Makrodatei für den XeLaTeX-Aufruf.

   Dazu sucht man nach `WinEdt/Exec/MiKTeX/pdflatex.edt` und macht auch davon eine Kopie in den Anwendungsdaten. Den Namen der Kopie ändert man in `xelatex.edt`. Anschließend öffnet man die Datei in WinEdt und ändert wiederum überall «`pdflatex`» in «`xelatex`».

3. Als Letztes braucht man einen Menü-Eintrag.

   Dazu geht man zu `Options->Menu Setup`, doppelklickt in dem Dialog auf `Accessories`, kopiert den Eintrag für `pdflatex`, fügt ihn dort ein, wo man ihn gerne hätte, und – wer hätte das gedacht – ändert «`pdflatex`» in «`xelatex`» sowohl im Titel als auch im Makroaufruf. Bildchen einfügen und Shortcuts festlegen kann man nach Geschmack.

Danach sollte man WinEdt schließen, neu öffnen und testen, ob alles funktioniert.

Diverses

Fallunterscheidungen mit `ifxetex`

Das Paket `ifxetex` [10] bietet den Schalter `\ifxetex...\else...\fi`. Benutzt man außerdem `\ifpdf` – was seltener nötig ist, als die meisten glauben – sollte man die Fallunterscheidung so aufbauen:

```
\usepackage{ifxetex}
...
\ifxetex
 %code nur für xetex
\else
 %code für pdf- + dvi-Output mit pdflatex/latex
 \ifpdf
 %code nur für pdf-Output mit pdflatex
 \else
 %code nur für dvi-Output
 \fi
\fi
```

Das Paket `xltxtra`

`xltxtra` [12] enthält eigentlich alle nötigen Pakete und Definitionen für einen guten Start mit XeLaTeX: Es lädt u. a. die Pakete `fontspec`, `xunicode`, `ifxetex` und `graphicx`. Es definiert die Befehle `\XeTeX` (XeTeX) und `\XeLaTeX` (XeLaTeX). Außerdem enthält es noch diverse sinnvolle Anpassungen.

Die Treiber anpassen

XeTeX verwendet (in MiKTeX) als Treiber für die `pdf`-Ausgabe `xdvipdfmx`. Dieser Treiber ist eine Variante von `dvipdfmx`, der wiederum eine Variante von `dvipdfm` ist. Im Zweifel oder wenn es nicht einen besseren, speziell für XeLaTeX entwickelten Treiber gibt, sollte man also einen Treiber für diese Programme benutzen.

Zuerst einmal sollte man aber überprüfen, ob die Pakete nicht von allein den richtigen Treiber verwenden, sofern ihnen der Anwender nicht ins Handwerk pfuscht, d. h. man sollte bei den Paketen und der Dokumentenklasse *alle* Optionen entfernen, die nach einer Treiberoption aussehen (`dvips`, `pdftex`,

dvipdfm usw.). Dann sollte man Schritt für Schritt überprüfen, welche Treiber die Pakete laden, indem man in die log-Datei schaut.

hyperref und graphicx z. B. sind bereits an XeTeX angepasst: Wenn XeTeX erkannt wird, lädt hyperref hdvipdfm.def, graphicx lädt xetex.def. beamer bringt allerdings den automatischen Modus von hyperref aus dem Tritt, daher muss man den richtigen Treiber per Option erzwingen:

```
\documentclass[hyperref=dvipdfmx]{beamer}
```

Bei anderen Paketen muss man gegebenenfalls den richtigen Treiber explizit angeben. Da empfiehlt sich ein Blick in die Dokumentation.

Papiergröße setzen

In XeTeX kann die Papiergröße durch \special{papersize=...} gesetzt werden. (Das ist die Syntax, die auch xdvipdfmx und dvips benutzen und verstehen.) XeTeX kennt auch die beiden pdfTeX-Befehle \pdfpagewidth und \pdfpageheight. Theoretisch sollte es also eigentlich einfach sein, die Papiergröße zu setzen: Ob mit dvips-\special oder pdfTeX-Befehlen, wichtig ist nur, dass sie überhaupt irgendwie gesetzt wird. Praktisch ist es etwas komplizierter, weil die meisten Pakete und Klassen sich große Mühe geben, automatisch die «richtige» Methode zu finden und sich damit bei XeLaTeX gelegentlich selbst ein Bein stellen.

Bei den KOMA-Script-Klassen sollte man die Option pagesize benutzen. Sie fügt den \special-Befehl ein und setzt auch \pdfpagewidth und \pdfpageheight auf die richtigen Werte. Damit tun die Klassen genau das Richtige für XeLaTeX.

Die Klasse memoir setzt ohne weiteres Zutun die richtige Papiergröße aber nur über den dvips-\special. Möchte man, dass auch \pdfpagewidth und \pdfpageheight die korrekten Werte erhalten, muss man dies manuell tun. Der Befehl \fixpdflayout funktioniert nicht mit XeLaTeX, da er weitere pdfLaTeX-spezifische Längen setzt.

geometry ist nicht völlig mit XeLaTeX kompatibel. Daher befindet sich in tex\xelatex\xetexconfig eine geometry.cfg, die geometry an XeLaTeX anpasst. Man sollte unbedingt die neueste Version dieser geometry.cfg benutzen, weil ältere Versionen einige unerwünschte Nebenwirkungen haben. Der Inhalt der neuen Version ist:

```
%% geometry.cfg for XeLaTeX - version of 2008-03-26

\@ifundefined{XeTeXversion}{}{%
 %% override to use the "pdftex" driver
 %% (i.e., \pdfpagewidth, \pdfpageheight) with XeTeX
 \def\Gm@checkdrivers{%
      \Gm@setdriver{pdftex}%
 }%
}%

\endinput

%% End of file 'geometry.cfg'.
```

Mit dieser `geometry.cfg` kann man dann bei allen drei Compilern – XeLaTeX, pdfLaTeX und LaTeX + dviXXX, wobei «dviXXX» der bevorzugte DVI-Treiber ist – denselben `geometry`-Aufruf `\usepackage[dviXXX]{geometry}` benutzen.

Bilder mit `graphicx` einfügen

Das Gespann `xelatex` und `xdvipdfmx` kommt mit ziemlich vielen Graphikformaten zurecht: Unterstützt werden u. a. die Formate `pdf`, `jpg`, `png`. Es ist nicht nötig, irgendwelche externen Hilfsdateien mit Boundingbox-Informationen zu erstellen. XeTeX kann die Größeninformationen aus den Bildern selbst auslesen. Wichtig ist aber, dass `graphicx` den Treiber `xetex.def` benutzt.

Auch `eps`-Bilder können benutzt werden[1]: `xdvipdfmx` erzeugt mithilfe des Ghostscripts von MiKTeX eine `pdf`-Version des Bildes, fügt sie in die `pdf`-Datei ein und löscht sie anschließend. Das funktioniert problemlos, verlangsamt aber natürlich die Kompilation. Bei vielen Bildern sollte man also besser die Bilder z. B. mit `epstopdf` dauerhaft konvertieren.

`xetex.def` erwähnt noch weitere Bildformate. Die komplette Liste der Dateiendungen lautet derzeit: `.pdf`, `.eps`, `.ps`, `.png`, `.jpg`, `.bmp`, `.pict`, `.tif`, `.psd`, `.mac`, `.sga`, `.tga`, `.gif`. Mir ist aber nicht bekannt, welche (außer den bereits erwähnten) in MiKTeX wirklich funktionieren. Meine Tests waren zumindest mit `gif`-und `bmp`-Bildern nicht erfolgreich.

[1] In anderen Betriebssystemen muss dies eventuell erst in `dvipdfmx.cfg` aktiviert werden.

Sprachumschaltung: Trennmuster

XeTeX benötigt wegen der Unicodeausgabe andere Trennmusterdateien als TeX/LaTeX, zumindest wenn die Sprache Sonderzeichen benutzt. Die `language.dat` bzw. der Tab «Language» in `MikTeX->settings` verweist daher oft nicht mehr direkt auf die Trennmusterdatei, sondern auf Dateien mit dem Prefix `xu-`. Diese Dateien kümmern sich dann um die nötige Fallunterscheidung. Ob für alle Sprachen die nötigen Trennmuster bereits existieren, ist mir nicht bekannt! Bei Problemen sollte man in der Mailingliste fragen.

Sprachumschaltung: `babel` und `polyglossia`

`babel` ist eigentlich nicht *ein* Paket sondern ein ganzes Bündel – und je nach Sprache verhält sich `babel` anders, manches davon funktioniert auch mit XeTeX, anderes beißt sich. Es kann daher keine allgemeine Empfehlung für oder gegen `babel` geben, sondern nur eine Reihe von Tipps und Anmerkungen:

- Die Definitionen der diversen Bezeichnungen (`\chaptername` usw.) sollten meistens korrekt sein, sofern `xunicode` geladen wurde.
- Bestimmte Definitionen muss man eventuell an die Kodierung von `fontspec` anpassen. Ein Beispiel zeigt das Listing auf Seite 22.
- Sobald eine fremde Schrift (wie z. B. bei den Sprachen `greek` oder `hebrew`) ins Spiel kommt, gibt es Probleme. Zum Setzen solcher fremden Schriften ändert `babel` die Kodierung in einer Weise, die sich nicht mit der Schriftauswahl von `fontspec` verträgt. Die Ausgabe wird zwar häufig dennoch richtig sein, aber die dabei verwendete Schrift ist wahrscheinlich nicht die gewünschte.

Die Alternative zu `babel` für XeLaTeX ist `polyglossia` [1]. Die Version 1.0 ist mittlerweile auf CTAN und kann auch mit dem Package Manager von MiKTeX installiert werden.

`polyglossia` ist natürlich gut an XeTeX angepasst und bestimmte Dinge löst es sehr elegant mit XeTeX-Befehlen. Beispielsweise muss `polyglossia` für die im Französischen nötigen Abstände zwischen Wörtern und Interpunktion nicht auf aktive Zeichen zurückgreifen, sondern verwendet `\XeTeXinterchartoks`. Andererseits ist `polyglossia` in vielen Bereichen noch ziemlich dünn. Die Sprachdateien enthalten oft nicht viel mehr als die Übersetzung der Standardnamen, viele der typographischen Feinheiten fehlen (noch).

Ein Teil des Codes von `babel` wird bereits während der Erstellung der Formatdateien über die Datei `hyphen.cfg` geladen. In den Betaversionen hatte auch `polyglossia` eine derartige `hyphen.cfg`: Aktiviert wurde sie, indem die Datei `hyphen_cfg` in `hyphen.cfg` umbenannt und an eine Stelle installiert wurde, wo sie bei der Formatherstellung für XeLaTeX vor der `hyphen.cfg` von `babel` gefunden wurde. Aber wenn man das tat, funktionierte natürlich bei diesem Format `babel` nicht mehr. Glücklicherweise gibt es dieses Problem bei der ersten offiziellen Version nicht mehr.

PSTricks und animate

PSTricks funktioniert zumindest teilweise: Man muss dazu die Dateien `pstricks.con` aus dem `xetex-pstricks`-Paket [5] in die Ordner `tex\xelatex\xetex-pstricks/` und `tex\xetex\xetex-pstricks/` oder in andere Ordner, die nur von XeLaTeX beziehungsweise XeTeX durchsucht werden, kopieren. Es ist wichtig, dass das normale LaTeX diese `pstricks.con` nicht zu sehen bekommt! Die `pstricks.con`-Dateien sind nur Wrapper-Dateien, die die Datei `xdvipdfmx.con`[2] laden. `xdvipdfmx.con` befindet sich mittlerweile in `tex\generic\pstricks`. Danach sollte man die FNDB (file name data base) aktualisieren, beziehungsweise für Linux `texhash` laufen lassen.

Anschließend wird `xdvipdfmx` die PSTricks-Bilder *on-the-fly* mit Ghostscript konvertieren und einfügen. Ebenso wie die Konvertierung von `eps`-Bildern verlangsamt dies die Kompilation. Mittlerweile klappt es auch in MiKTeX (in früheren Versionen wurden bei den temporären Dateien die Pfadtrenner verschluckt), daher konnte ich einige kurze Tests durchführen. Für mich etwas überraschend funktionierte auch PSTricks-Code, der sich nicht innerhalb einer `pspicture`-Umgebung befindet. Es ist beispielsweise möglich, zwei Knoten im laufenden Text mit einer Linie zu verbinden. Anfangs gab es Schwierigkeiten bei 3D-Effekten: Beim Dodekaeder-Kalender (`\psCalDodecaeder`) aus dem `pst-calendar`-Paket z. B. erzeugte `xdvipdfmx` einen Haufen temporärer Bilder (und brauchte dafür *sehr* lange), war aber am Ende offensichtlich nicht in der Lage, die Teile korrekt zu verzerren und zusammenzufügen: Man erhielt schlussendlich diverse übereinander gestapelte Fünfecke. Mittlerweile klappt es mit dem Kalender (es dauert aber immer noch sehr lange).

[2] Sie hieß zuerst `xdvipdfmx.con`, dann zwischendurch mal `xetex-pstricks.con` und jetzt wieder `xdvipdfmx.con`

Die neueste Version von `animate` unterstützt nun `(x)dvipdfmx` und damit auch XeLaTeX. Sie benötigt dazu aber eine aktuelle Version von `(x)dvipdfmx`, die eine neue Syntax (`\special{pdf:stream...}`) versteht.

`\MakeUppercase`-Probleme

`\MakeUppercase{ß}` funktioniert mit XeLaTeX nicht korrekt: Das Ergebnis ist nicht «SS» sondern wieder ein «ß». Man muss daher dort, wo dies von Bedeutung sein könnte, `\ss` (oder `"s` bei `(n)german`) verwenden.

Was mit XeTeX nicht geht

XeTeX hat keine der `microtype`-Fähigkeiten. Das finde ich besonders schade.

Die Eingabe oder: Was ist mit `inputenc`?

In Kürze:
- `inputenc` sollte man *nicht* mit XeLaTeX verwenden.
- `xunicode` sollte man benutzen – es definiert diverse Textbefehle. Das Paket sollte immer *nach* `fontspec` geladen werden.
- Nach Möglichkeit sollte die `tex`-Datei in UTF-8 kodiert sein. «Normale» 8-Bit-Textkodierungen gehen auch, die Kodierung muss aber korrekt deklariert werden.
- Mit XeLaTeX haben Pakete wie `listings`, `soul` und `url` keinerlei Probleme mit Umlauten und UTF-8-Dateien.

(pdf)TeX ist intern 8-Bit-orientiert. XeTeX baut intern auf Unicode[3] auf. Beide TeX-Varianten können bei der Eingabe mit 8-Bit-Textdateien und UTF-8-Textdateien umgehen, aber die Verarbeitung unterscheidet sich in manchen Punkten erheblich. (XeTeX kann auch UTF-16LE und UTF-16BE-Eingabedateien verarbeiten, aber darauf werde ich nicht weiter eingehen.)

Da ich immer mal wieder feststelle, dass eine ganze Menge TeX-Benutzer nur eine ziemlich vage Vorstellung haben, was eigentlich 8-Bit-Textkodierungen und UTF-8 sind und wie dieses ganze `inputenc`-System eigentlich funktioniert,

[3] Genauer: auf UTF-16 (und nicht UTF-8 oder UTF-32) – aber das macht nur bei richtig exotischen Zeichen einen Unterschied.

folgt hier einmal eine kleine Einführung in 8-Bit und UTF-8 und wie die normalen (pdf)TeX-Varianten damit umgehen.

8-Bit-Textkodierungen

Jede Datei ist einfach eine lange Folge aus Nullen und Einsen («Bits»): `1001001100111111101010101111...`. Das gilt auch für Textdateien.

Wenn man eine derartige Datei in einem Editor wie WinEdt öffnet, sieht man aber keine Folgen von Nullen und Einsen sondern Buchstaben, Leerzeichen, Zeilenumbrüche; d. h. der Editor übersetzt die Bitfolge in «sinnvolle» Zeichen. 8-Bit-orientierte Editoren wie WinEdt zerlegen dazu die Bitfolge in Päckchen je 8 Bit («Oktetts»). Jedes dieser Oktetts entspricht einem bestimmten Zeichen, das dann angezeigt wird. (Das passiert beim Öffnen *jeder* Datei, auch wenn man eine `pdf`, `exe` oder eine `jpg` öffnet. Nur sieht das Übersetzungsergebnis dann nicht sehr sinnvoll aus.)

$$\underbrace{11100100}_{\text{ä}}\underbrace{01100001}_{\text{a}}\ldots$$

Es gibt insgesamt 256 verschiedene 8-Bit-Päckchen, daher wird der Editor maximal 256 verschiedene Zeichen bei der Anzeige der Datei verwenden. Die eine Hälfte der Oktetts – all die Oktetts, die mit einer Null beginnen – ist dabei ziemlich unproblematisch: Das sind die ASCII-Zeichen und unabhängig von Betriebssystem oder Sprachumgebung wird fast jeder Editor sie gleich anzeigen[4]. Bei der anderen Hälfte gibt es Unterschiede: `11100100` wird ein Windows-Editor als `ä` anzeigen, auf dem Mac wird die Bitfolge als ‰ interpretiert:

abc äöü abc ‰ˆ¸

```
\inputencoding{ansinew}
abc äöü
\inputencoding{applemac}
abc äöü
```

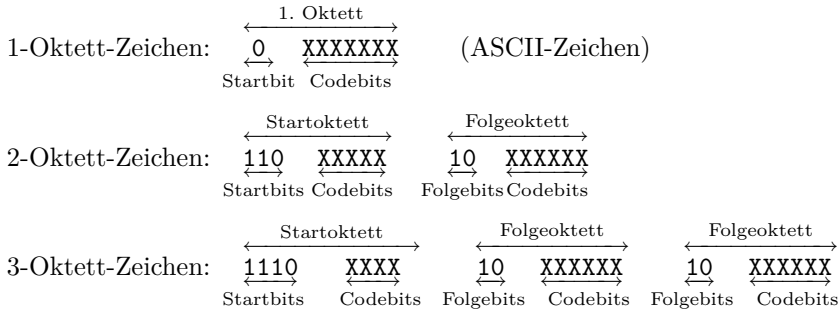
Beim Tippen und Speichern eines Textes mit einem 8-Bit-Editor passiert dann das Entsprechende: Es können maximal 256 Zeichen eingeben werden und jedes dieser Zeichen wird in der Datei als ein Oktett gespeichert. Die ASCII-Zeichen werden überall in dieselben, mit Null beginnenden 8-Bit übersetzt, dagegen wird `ä` in Windows zu `11100100` und im Mac zu irgendwas anderem.

[4] In Japan benutzt man angeblich ein ¥ anstelle des \. Außerdem gibt es noch Unterschiede, wie der Zeilenumbruch kodiert wird.

(pdf)TeX ist, wie bereits gesagt, 8-Bit-orientiert. Wie ein 8-Bit-Editor zerlegt es einen Eingabestrom in Oktetts. Es kann also maximal 256 Eingabezeichen unterscheiden. Wenn ein Eingabeoktett mit einer Null beginnt, geht (pdf)TeX davon aus, dass das entsprechende ASCII-Zeichen gemeint ist. Falls die Datei Oktetts außerhalb des ASCII-Bereichs enthält, braucht pdfTeX weitere Informationen. In LaTeX wird dazu das Paket `inputenc` benutzt: Das Paket «aktiviert» alle Nicht-ASCII-Zeichen, d. h. macht sie zu Befehlen. Anschließend kann diesen Befehlen eine sinnvolle Definition gegeben werden. (Nicht so sinnvolle Definitionen sind auch möglich: z. B. würde `\DeclareInputText{128}{\,DM}` in einem in cp1252-kodierten Text jedes Eurozeichen durch den «DM» ersetzen.)

UTF-8

Benutzt ein Editor UTF-8, wird ein Zeichen nicht in eine Bitfolge mit einer festen Länge übersetzt, sondern es werden für ein Zeichen ein oder mehrere Oktetts benutzt. Erlaubt sind derzeit bis zu vier Oktetts, wobei aber die 4-Oktett-Zeichen selten benötigt werden. ASCII-Zeichen sind beispielsweise durch kurze 1-Oktett-Folgen kodiert, Umlaute wie «ä» oder «ü» sind 2-Oktett-Zeichen. Zur Kennzeichnung der Länge und der Bestandteile eines Zeichens benutzt UTF-8 sogenannte Start- und Folgebits:



Es ist wichtig, sich folgende Dinge zu merken:

- Bei einer Textdatei, die nur ASCII-Zeichen enthält, ist es unwichtig, ob man sie als UTF-8 oder in einer beliebigen 8-Bit-Kodierung speichert. Das Ergebnis ist völlig identisch. Es ist auch egal, ob man sie in einem UTF-8- oder 8-Bit-Editor öffnet: Beide werden das Gleiche anzeigen. Das ist ziemlich beruhigend, weil es bedeutet, dass XeTeX mit den meisten LaTeX-Styles keine Probleme haben wird.

- In UTF-8 sind bestimmte Oktettkombinationen nicht erlaubt bzw. sie können einfach nicht vorkommen: Auf ein ASCII-Zeichen kann z. B. kein Oktett folgen, das mit `10` beginnt. Solche Kombinationen sind in 8-Bit-Textdateien aber gang und gäbe.

Wie (pdf)TeX UTF-8-Dateien verarbeitet

Wenn (pdf)TeX eine UTF-8-kodierte Datei bearbeitet, dann tut es einfach das, was es immer macht: Es zerlegt den Eingabestrom in Oktetts.

Das heißt, auch wenn die Datei in einem Editor so aussieht, als enthielte sie tausende verschiedene Zeichen, *(pdf)TeX unterscheidet nur 256 Oktetts*! Ein UTF-8-«ä» (`11000011 10100100`) sind für (pdf)TeX nicht *ein* sondern *zwei* Eingabezeichen[5].

Benutzt man `inputenc` mit der Option `utf8`, macht es die Startoktetts zu Befehlen. Diese Befehle werden dann so definiert, dass sie mit den Folgeoktetts als Argument eine sinnvolle Ausgabe erzeugen. Das funktioniert ganz gut, stößt aber auf Schwierigkeiten, wenn andere Befehle die Eingabe ebenfalls zeichenweise (also Oktett für Oktett) verarbeiten. Pakete, die mit Umlauten in UTF-8-Dateien Probleme haben, sind z. B. `listings`, `url` und `soul`. Zu letzterem gibt es `soulutf8` als Alternative.

Eingabekodierung bei XeTeX

XeTeX baut auf Unicode auf. Es ist nicht wie TeX auf 256 Eingabezeichen beschränkt, sondern kann mit den vielen tausenden Unicodezeichen umgehen. Es zerlegt eine UTF-8-Datei nicht in Oktetts oder ähnliches: Jedes Unicodezeichen in der Datei ist für XeTeX *ein* eigenes, ganz normales Eingabezeichen. Wenn eine Eingabedatei in UTF-8 oder UTF-16 kodiert ist, braucht man daher nichts weiter zu tun. XeTeX wird die Eingabe korrekt interpretieren.

XeTeX kennt auch viele der üblichen 8-Bit-Kodierungen. 8-Bit-kodierte Dateien werden beim Einlesen in das interne Unicodeformat konvertiert. Pakete wie `inputenc` oder `ucs` werden daher bei XeLaTeX nicht benötigt – schlimmstenfalls schaden sie.

Wenn eine Eingabedatei 8-Bit-kodiert ist, muss man die Kodierung deklarieren. Dazu gibt es zwei neue primitive Befehle: `\XeTeXinputencoding` und `\XeTeXdefaultencoding`. Beide Befehle benötigen ein Argument, den Namen

[5] Zeichen hat übrigens nichts mit `\token` zu tun.

der Kodierung. Das Argument wird einfach durch ein Leerzeichen getrennt hinter den Befehl geschrieben und *mit einem Leerzeichen oder* `\relax` *beendet*. Die für LaTeX typischen Klammern ergeben keinen Fehler, haben aber auch nicht ihre übliche Funktion als Begrenzung eines Arguments. Die Dokumentationen enthalten keine genauen Angaben, welche Namen für die Kodierungen benutzt werden können. Angeblich funktionieren all die Namen, die in Internetseiten und von E-Mail-Programmen benutzt werden. Für Windows funktioniert jedenfalls `cp1252`. Der Name «`auto`» ergibt das Normalverhalten: Dateien werden als UTF-8/16 gelesen. Wenn XeTeX einen Namen nicht kennt, gibt es in älteren XeTeX-Versionen eine Meldung auf dem Bildschirm aus, aber keinerlei Hinweis in der `log`-Datei. In neueren XeTeX-Versionen ist es umgekehrt.

`\XeTeXinputencoding` deklariert die Kodierung für die *aktuelle* Datei von dem Punkt an, an dem der Befehl auftritt. Der Befehl kann überall benutzt werden – sinnvoll ist er natürlich irgendwo *vor* dem ersten Nicht-ASCII-Zeichen – und auch beliebig oft. Der folgende Code hat eine ähnliche Wirkung wie der Code in dem Beispiel auf Seite 17:

```
\XeTeXinputencoding cp1252
abc äöü
\XeTeXinputencoding mac
abc äöü
```

Die Wirkung von `\XeTeXinputencoding` endet mit dem Ende der aktuellen Gruppe oder spätestens am Ende der aktuellen Datei. `\XeTeXinputencoding` hat keinen Einfluss auf Dateien, die nach der Deklaration mit `\input` usw. eingelesen werden.

`\XeTeXdefaultencoding` hat keine Wirkung auf die aktuelle Datei, deklariert aber die Kodierung für alle Dateien, die anschließend eingelesen werden. Trotz des `default` im Namen eignet sich der Befehl *nicht*, um die Kodierung für ein ganzes Projekt festzulegen. Denn alle Dateien, die von XeTeX geschrieben werden (`aux`, `toc` usw.) sind UTF-8 kodiert – da wäre es nicht so gut, sie als 8-Bit einzulesen. Nützlich ist `\XeTeXdefaultencoding` festzulegen, um die Kodierungen von Inputdateien festzulegen, ohne sie selbst zu verändern:

```
\XeTeXdefautencoding cp1252
\input{file} %8-Bit-Datei
\XeTeXdefautencoding auto
```

Größere Projekte im 8-Bit-Modus mit XeLaTeX zu bearbeiten, kann also ein bisschen mühselig werden, da an vielen Stellen die Kodierung deklariert werden muss. Man sollte sich also überlegen, ob ein Umstieg auf UTF-8 sinnvoll ist. Dabei sollte man nicht vergessen, dass der Umstieg auch Nachteile haben kann, wenn man weiterhin auch LaTeX benutzen will (wie man am Beispiel von `listings` sehen kann).

Wenn man sich bei der Eingabekodierung vertut

Wenn XeTeX im «`auto`»-Modus eine 8-Bit-kodierte Eingabedatei bearbeitet, die Sonderzeichen wie z. B. Umlaute enthält, wird XeTeX höchstwahrscheinlich ziemlich schnell über in UTF-8 nicht mögliche Oktettkombinationen stolpern. In älteren Versionen ist XeTeX dann einfach abgestürzt. Mittlerweile erhält man eine Warnung in der `log`-Datei:

```
Invalid UTF-8 byte or sequence at line 4 replaced by U+FFFD.
```

XeTeX fährt danach mit der Bearbeitung im sogenannten «Bytes-Modus» fort. Wenn die Sonderzeichen sich nur in Kommentaren befinden und der Code selbst reines ASCII ist, kann man die Warnung getrost ignorieren; im anderen Fall wird die Ausgabe aller Wahrscheinlichkeit nach fehlerhaft sein.

Wenn man eine UTF-8-Datei versehentlich als 8-Bit-Datei deklariert und bearbeitet, wird die Ausgabe wahrscheinlich auch fehlerhaft sein. Es wird aber keinerlei Warnungen oder ähnliches geben (können).

Achtung: Fehlende oder falsche Zeichen in der Ausgabe gehen zwar oft, aber nicht immer, auf eine falsche Deklaration der Eingabekodierung zurück. Die Schrift kann auch schuld sein.

UTF-8 Editoren in Windows

Im Gegensatz zu Mac und Linux ist UTF-8 in Windows kein Standard. UTF-8-Editoren sind noch dünn und TeX-UTF-8-Editoren noch dünner gesät.

TeXniccenter (`http://www.texniccenter.org/`) kann meines Wissens gar kein UTF-8.

WinEdt (`http://www.winedt.com`) kann ein bisschen UTF-8: Solange eine UTF-8-Datei nicht mehr als die in einer 8-Bit-Datei üblichen 256 Zeichen benutzt, kann WinEdt diese Datei lesen und auch als UTF-8 wieder speichern, d. h. WinEdt kann UTF-8-Dateien mit normalen «westeuropäischen» Texten

bearbeiten; mit Texten, die sowohl deutsche Umlaute als auch griechische Buchstaben enthalten, kann es nicht umgehen.

Texmaker (`http://www.xm1math.net/texmaker/`) und Winshell (`http://www.winshell.org/`) können laut Beschreibung Unicode. Wie gut sie mit UTF-8 umgehen und wie gut sie zusammen mit MiKTeX funktionieren, kann ich nicht sagen.

Emacs beherrscht natürlich UTF-8, arbeitet aber leider nicht so gut mit MiKTeX zusammen wie WinEdt: z. B. werden `pdf`-Dateien nicht automatisch vor der Kompilation geschlossen.

Zeichenbefehle mit `xunicode`

In LaTeX werden sehr viele Zeichen über Befehle eingegeben. Beispiele sind `\$`, `\textbackslash`, `\textyen`, `\"a`. Oft hängt die Definition derartiger Befehle von der Kodierung der Ausgabeschrift ab. In einer OT1-kodierten Schrift z. B. ist das «ß» an der Position 25, in T1 an der Position 255. Daher enthält `ot1enc.def` für `\ss` eine andere Definition als `t1enc.def`:

```
\DeclareTextSymbol{\ss}{OT1}{25}
\DeclareTextSymbol{\ss}{T1}{255}
```

In XeTeX kommen nun zu den OT1-, T1-, T2A- usw. kodierten Schriften noch die Unicode-kodierten Schriften hinzu. Also braucht man einen weiteren Satz solcher Definitionen. Diesen stellt das Paket `xunicode` [8] zur Verfügung. Als Kodierungsname wird dabei der von dem Paket `fontspec` geprägte Name EU1 benutzt. (Und von daher gehören die Definitionen eigentlich in die Datei `eu1enc.def`. Derzeit muss man sie aber noch separat laden.) `xunicode` sollte immer *nach* `fontspec` geladen werden.

`xunicode` muss sehr viel mehr Befehle definieren als z. B. `t1enc.def`, denn Unicode enthält nun mal sehr viel mehr Zeichen als die 256-Zeichen-Kodierung T1. An diversen Stellen in der `sty`-Datei findet man Fragezeichen, weil der Autor noch keine endgültige Entscheidung über die «richtige» Definition getroffen hat. Man sollte also immer damit rechnen, dass es zu Änderungen kommt!

Man sollte auch damit rechnen, dass einzelne Befehle nicht von `xunicode` erfasst werden und erst nachbearbeitet werden müssen, bevor sie korrekt mit XeLaTeX funktionieren. Die Anführungszeichen `\glqq` und `\grqq` beispielsweise benötigen folgenden Code:

```
\usepackage[ngerman]{babel}
\ProvideTextCommand{\glqq}{EU1}{%
  \textormath{\quotedblbase}{\mbox{\quotedblbase}}}
\ProvideTextCommand{\grqq}{EU1}{%
  \textormath{\textquotedblleft}{\mbox{\textquotedblleft}}}
```

Ligaturen

Einige Zeichen in der Ausgabe kann man über Ligaturen erzeugen. Beispiele sind `--` für –, `''` für ”, `?`` für ¿ aber auch `fl` für fl.

Ligaturen sind keine Befehle, sondern eine Eigenschaft der Schrift: Die zugehörigen Anweisungen stehen in der `tfm`-Datei. Wie sie in XeTeX funktionieren, wird daher weiter unten beschrieben.

Schriften mit XeLaTeX: Die alte Methode per `tfm`-Datei

In LaTeX werden Schriften intern durch die Angabe des Namens der `tfm`-Datei geladen. Dazu wird der primitive TeX-Befehl `\font` benutzt:

`\font\myname =` ⟨*`tfm`-Name*⟩ ⟨*TeX-Schriftargumente*⟩

LaTeX sucht dann diese `tfm`-Datei, entnimmt ihr die nötigen metrischen Informationen, setzt damit die Seite, schreibt einen Verweis auf die `tfm`-Datei in die `dvi`-Datei und überlässt es dem Treiber, die dazu gehörige «richtige» Schrift zu finden und einzufügen.

Dies funktioniert auch mit XeLaTeX: Dieses kann genauso wie LaTeX vorhandene `tfm`-Dateien öffnen und nutzen. Die Suchpfade sind die üblichen. Theoretisch funktionieren daher alle Schriftpakete wie `fontenc`, `fourier`, `mathptmx` usw. auch mit XeLaTeX.

Praktisch gibt es aber drei Probleme:

1. Mit einigen Schriften haben die Treiber anschließend Probleme: Der Treiber `xdv2pdf` kann nicht mit virtuellen Schriften umgehen, wird also bei den meisten etwas aufwendigeren Schriften versagen. `xdvipdfmx` mag keine `pfa`-Dateien, die müssen also zuerst mit z. B. `t1binary` in `pfb`-Dateien konvertiert werden. Als `map`-Datei benutzt `xdvipdfmx` diejenige von `dvipdfmx`.

2. Die Eingabe funktioniert nicht wie gewohnt. Dies will ich an einem Beispiel erläutern: Im T1-Encoding befindet sich das «ß» an der Position 255 (dezimal) bzw. FF (hexadezimal). Die normale Verarbeitung in LaTeX mit `inputenc` läuft so:

   ß $\xrightarrow{\texttt{inputenc}}$ `\ss` $\xrightarrow{\texttt{t1enc.def}}$ `\char"FF`

   In XeLaTeX fehlt nun der erste Teil, da `inputenc` ja nicht verwendet werden sollte. Das «ß» wird also einfach «durchgereicht»:

   ß = Unicode U+00DF $\longrightarrow$ `\char"DF` = SS.

   Nicht alle Sonderzeichen werden falsch sein, denn bei einigen stimmt die Unicode- mit der T1-Position überein. Dennoch empfiehlt es sich, die alte ASCII-Eingabemethode wieder zu üben, wenn man in XeLaTeX Schriften über LaTeX-Pakete und -Encodings nutzen möchte. Und ganz besonders sollte man sich verkneifen, Zeichen aus den höheren Unicodebereichen zu benutzen.

3. Worttrennungen können falsch oder anders sein, wenn man Schriften benutzt, die eine LaTeX-Kodierung (also T1, OT1, T2A usw.) verwenden, weil die neuen Trennmusterdateien von XeTeX von Unicode-Schriften ausgehen. Dies ist natürlich nur bei Textschriften von Bedeutung.

Systemschriften nutzen

Die wirkliche Stärke von XeTeX liegt aber darin, dass XeTeX genau wie andere Programme während der Übersetzung der `tex`-Datei direkt auf die «richtigen» Schriften zugreifen kann und ihnen Informationen über die Größen der Zeichen, Ligaturen und vieles mehr entnehmen kann. Insbesondere braucht XeTeX keine externe `tfm`-Datei mehr. XeTeX kann mit Type1-, TrueType- und OpenType-Schriften umgehen, aber die meisten Möglichkeiten (und die wenigsten Probleme) hat man mit letzteren.

Intern benutzt XeTeX auch bei diesen Schriften den primitiven `\font`-Befehl, dessen Syntax dafür erweitert wurde:

`\font\myname = "`⟨*Schriftname*⟩`:`⟨*Features*⟩`" `⟨*TEX-Argumente*⟩

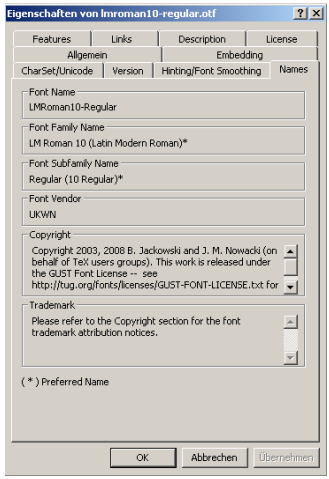
Wesentlich sind die Anführungszeichen um den Schriftnamen: Daran erkennt XeTeX, dass der Schriftname sich auf eine «wirkliche» Schrift bezieht[6]. Der

[6] Eine Erläuterung dazu findet man unter `http://tug.org/mailman/htdig/xetex/2007-July/007048.html`.

Doppelpunkt und die ⟨*Features*⟩ sind (wie die ⟨*TEX-Schriftargumente*⟩) optional. Damit die Schrift auch wirklich genutzt wird, muss natürlich der *Schriftname* stimmen und XeTeX muss die Schrift finden.

Vorbereitung

Windowsbenutzer sollten unbedingt die «Font properties extension» installieren [2]. Nach Installation dieses Tools erhält man deutlich mehr und nützlichere Informationen, wenn man den Punkt «Eigenschaften» im Kontextmenü einer Schrift aufruft. Das hilft ungemein bei der Bestimmung des korrekten Schriftnamen:



Schriften finden: `fc-cache`

Wenn XeTeX während der `tex->xdv`-Kompilation direkt auf die Schriften zugreifen will bzw. muss, um die `tfm`-Informationen und anderes zu extrahieren, benutzt es nicht die Standardsuchpfade, sondern durchsucht die Ordner, die in `<UserConfig>\fontconfig\config\localfonts.conf` und `<UserConfig>\fontconfig\config\localfonts2.conf` eingetragen sind. Die zweite Datei ist für Benutzereinträge gedacht. In `localfonts.conf` steht u. a. der Standardschriftenordner `C:\Windows\Fonts`, was bedeutet, dass alle «im System installierten» Schriften zur Verfügung stehen.

XeTeX benutzt für die Suche nach Schriften einen Cache. Im Allgemeinen aktualisiert MiKTeX den Cache, wenn man Pakete installiert. Mit dem Kommandozeilenprogramm `fc-cache` kann man es auch selbst tun. Die Option `--help` listet die möglichen Optionen auf. Meine (wenigen) Versuche haben ergeben, dass `fc-cache` ausreicht, wenn man einen neuen Ordner in `localfonts.conf` eingetragen hat. Wenn aber in einen Ordner, der bereits im Cache enthalten ist, neue Schriften kopiert wurden, sollte man `fc-cache -f` aufrufen.

Leider verlässt sich XeTeX nicht völlig auf den Cache: Wenn man versucht, eine nicht existierende Schrift zu nutzen (z. B. wenn man sich beim Namen vertippt), dann startet XeTeX eine längere Suche[7] auf der Festplatte, bevor es mit einer Fehlermeldung sein Scheitern zugibt.

Schriftnamen

Die gewünschte Schrift kann man auf verschiedene Weise benennen. Die im Folgenden benutzten Beispiele entstammen der angegebenen Abbildung zur Latin-Modern-Schrift, beispielsweise:

- Als *external location* durch Angabe des Dateinamens (wenn nötig inklusive des Pfades). Dieser muss dann in eckige Klammern gesetzt werden. Beispiel: `[lmroman10-regular]`. Das Paket `fontspec` benutzt die Option `ExternalLocation` zur Kennzeichnung solcher Namen.
- Durch Angabe des internen Namens der Schrift, also zum Beispiel: `LMRoman10-Regular`. Das ist oft (aber aus irgendeinem Grunde nicht immer) der Name, den man in den üblichen Schriftmenüs in Windowsprogrammen sieht. Im Eigenschaftenmenü der Schrift steht dieser Name unter *Font Name*. Korrekte Groß- und Kleinschreibung ist wichtig! Die Namen enthalten häufig Leerzeichen; das ist meistens unproblematisch, aber in Befehlen wie `\DeclareFontShape` muss man sie mit `\space` eingeben, sonst werden sie verschluckt.
- Durch Angabe des Familiennamens, des *Font Family Name*, und eventuell zusätzlich noch des *Font Subfamily Name*. Dabei sollte man die als «Preferred Names» gekennzeichneten Namen nehmen; die anderen sind nicht geeignet. Beispiele: `Latin Modern Roman` oder `Latin Modern Roman 10 Regular` oder `Latin Modern Roman 10 Bold`, letzteres ergibt dann die Fettschrift. Der Familienname ist, wenn es ihn gibt, meist die beste Lösung. Im Falle

[7] Bei mir dauert sie über eine Minute; das ist in Computerzeit fast eine Ewigkeit.

der Latin-Modern-Schriften z. B. benutzt XeTeX dann für alle Schriftgrößen den passenden optischen Schriftgrad, also `lmodern5-regular.otf` bei kleinen Schriften.

Ligaturen

XeTeX (und das Paket `fontspec`) unterscheidet bei Ligaturen zwei Dinge: Das sind zum einen die «richtigen» typographischen Ligaturen also z. B. «fi» für «fi». Ob und in welchem Umfang eine Schrift derartige Ligaturen enthält, ist eine Entscheidung des Schriftdesigners. Diese Ligaturen werden bei den direkt geladenen Schriften auch konsequenterweise über die *font features* gesteuert. `fontspec` hat dafür die Option `Ligatures`. Zum anderen sind da aber auch die TeX-spezifischen Ligaturen wie `--`. Das sind nicht wirklich typographische Ligaturen, sondern sie sollen nur die Eingabe vereinfachen. Wer solche Eingabemethoden benutzt, will sie wahrscheinlich bei jeder Schrift verwenden. Und da nicht zu erwarten ist, dass alle Schriftdesigner solche TeX-Eigenheiten in ihre Schriften einbauen, muss man sie von außen aufdrücken. Dies geschieht in XeTeX über sogenannte `mapping`-Direktiven. Die dafür benötigten Daten stecken in Dateien mit der Endung `.tec`. Das sind binäre Dateien, die mit einem Programm namens `TECkit-Compile` aus Quelldateien (mit der Endung `.map`) erzeugt werden. Auf der Homepage von XeTeX gibt es das Programm und eine Anleitung. Aber für die üblichen TeX-Ligaturen ist die Datei bereits vorhanden. Mit `fontspec` aktiviert man sie mit der Option `Mapping`. Beim primitiven `\font`-Befehl geht es so:

```
\font\x = "Latin Modern Roman:mapping=tex-text"
```

Zeichen und Symbole ausgeben

Die meisten Zeichen erhält man einfach dadurch, dass man das entsprechende Zeichen bei der Eingabe benutzt. Da die Eingabe und die meisten Schriften sich an Unicode orientieren, funktioniert das wunderbar und ist besonders für Autoren, die in fremden Schriften schreiben, eine ungeheure Erleichterung. Aber oft kann oder will man ein Zeichen nicht direkt, sondern über einen Befehl eingeben. Dafür gibt es – neben den vielen Befehlen, die Pakete wie `xunicode` bereitstellen – den auch in TeX vorhandenen Befehl `\char`⟨*Zahl*⟩ (und die LaTeX-Version `\symbol{`⟨*Zahl*⟩`}`) und `\XeTeXglyph`⟨*Zahl*⟩. ⟨*Zahl*⟩ kann bei allen Befehlen in hexadezimaler (beginnend mit `"`), oktaler (beginnend mit `'`) oder dezimaler Notation gegeben werden. In TeX akzeptiert der `\char`-Befehl nur Werte kleiner als 256, größere erzeugen einen Fehler, in XeTeX

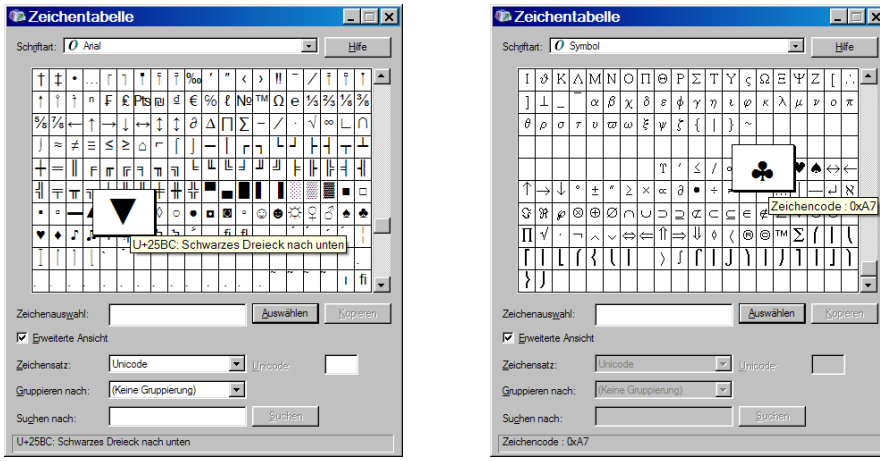


Abbildung 1: Links: Zeichentabelle einer mit Unicode kodierten Schrift. Rechts: Keine Unicode-Kodierung

scheint es keine Obergrenze zu geben – zumindest habe ich sie noch nicht gefunden.

Je nach Schriftformat unterscheiden sich Wirkung und Anwendbarkeit von `\char` und `\XeTeXglyph`.

Lädt man die Schrift mit der alten Methode über die `tfm`-Datei, kann man nur `\char` benutzen. `\XeTeXglyph` erzeugt eine Fehlermeldung. Sinnvoll sind in diesem Fall natürlich nur Werte zwischen 0 und 255, bei größeren Werten gibt es keine Ausgabe.

Lädt man die Schrift mit der neuen XeTeX-Methode, hängt viel vom Typ, genauer von der internen Kodierung, der Schrift ab. Die Abbildung 1 zeigt zwei Zeichentabellen. Die linke Schrift benutzt Unicode. Man erkennt das an dem «U+» beim Zeichencode. Die rechte Schrift benutzt keine Unicode-Kodierung, dort ist auch ein Teil der Auswahlmenüs ausgegraut. (Man sollte aber nicht glauben, dass alle Schriften, die behaupten, dass sie Unicode benutzen, sich auch an den Unicodestandard halten. Ich besitze beispielsweise einige Schachschriften, die für die diversen Brettzeichen die Codes normaler Buchstaben wie K oder D statt der dafür gedachten Codes in der sogenannten «Private Use Area» (U+E000 bis U+F8FF) benutzen.

Bei Schriften, die intern in Unicode kodiert sind, kann man alle verfügbaren Zeichen mit `\char`⟨*Unicode*⟩, also z. B. `\char"E001`, erhalten. Wenn eine Schrift das Zeichen nicht enthält, steht meist ein kleiner Platzhalter in der `pdf`-Datei.

Bei Nicht-Unicode-Schriften ist die Lage etwas komplizierter: `\char` funktioniert da manchmal einfach nicht – es gibt zwar keine Fehlermeldungen, aber auch keine Ausgabe. Dann muss man auf `\XeTeXglyph`⟨*slot*⟩ zurückgreifen. Dieser Befehl gibt das Zeichen aus, das sich an dem durch das Argument angegebenen *slot* befindet. Das Ergebnis ist *sehr* schriftspezifisch! Man erhält einen Fehler, wenn der Wert von *slot* zu groß ist. Daher empfiehlt es sich, zuerst die Anzahl der Glyphen mit `\XeTeXcountglyphs` zu bestimmen. Das folgende Listing zeigt, wie man mit einer Schleife alle Zeichen in der Schrift «Symbol» ausgeben kann – dabei ist besonders die Position 177 interessant. In der Windows-Zeichentabelle ist dort eine leere Stelle, die Ausgabe von XeLaTeX zeigt dagegen den abgebissenen Apfel von Apple – ein klarer Fall von Windows-Zensur `;-)`.

```
\documentclass{article}
\usepackage{ifthen}

\begin{document}%
\raggedright
\newcounter{glyphcount}
\setcounter{glyphcount}{0}
\font\myfont = "Symbol"
\whiledo
 {\value{glyphcount}<\XeTeXcountglyphs\myfont}
 {\arabic{glyphcount}:~%
  {\myfont\XeTeXglyph\arabic{glyphcount}}\quad
  \stepcounter{glyphcount}}
\end{document}
```

Wenn Glyphen fehlen

Die vielen relativ kleinen Zeichensatztabellen von LaTeX haben einen großen Vorteil: LaTeX weiß im Allgemeinen, wenn sich eine Schrift für die aktuelle Zeichensatztabelle nicht eignet und nimmt dann eine geeignete Substitution vor. Der Aufruf `{\fontfamily{skaknew}\selectfont Kaffee}` beispielsweise führt zu folgender Meldung in der `log`-Datei:

```
LaTeX Font Warning: Font shape ‘T1/skaknew/m/n’ undefined
```

```
(Font)          using 'T1/cmr/m/n' instead
```

Was auf Deutsch heißt: «Tut mir leid, aber mit der Schrift «skaknew» kann man keine westeuropäischen Texte schreiben, ich werde daher auf «cmr» ausweichen.» Und damit hat LaTeX auch völlig recht, denn «skaknew» ist eine Schachschrift.

Einen derartigen Substitutionsmechanismus gibt es in XeTeX nicht. Das Paket `fontenc` definiert zwar die Kodierung `EU1`, aber nur aus formalen Gründen. Die Kodierung wird einfach bei allen Schriften benutzt. Derzeit gibt es daher keinerlei Prüfung, ob eine Schrift das aktuell angeforderte Zeichen überhaupt enthält. Und es gibt auch keine Warnungen oder Fehlermeldungen, wenn das nicht zutrifft. Dabei geht es nicht nur um eindeutig «exotische» Zeichen, bei denen man damit rechnet, dass sie in der Schrift nicht vorhanden sind. Auch «harmlose» Akzentbefehle wie `\b` oder `\d` können betroffen sein.

Allerdings enthält XeTeX diverse Befehle, mit denen man Eigenschaften der Schrift, also z. B. welche Skripte («Schriftsprachen») sie unterstützt oder welche Glyphen sie enthält, abfragen kann. Es ist also nicht ausgeschlossen, dass irgendwann auch XeTeX auf eine andere Schrift ausweicht, wenn die aktuelle Schrift das gewünschte Zeichen nicht enthält. Aber bei Kraut-und-Rüben-Schriften, die sich an keinerlei Standards orientieren wie z. B. die vielen Schachschriften, wird das auch nicht helfen.

Suchen und Kopieren in pdf-Dateien

Zeichen aus Nicht-Unicode-Schriften wie der oben erwähnten Schrift «Symbol» kann man natürlich nicht kopieren – da kann irgendetwas herauskommen.

Bei Schachschriften, die für ihre Symbole den Unicode von normalen Buchstaben benutzen, erhält man erwartungsgemäß beim Kopieren diese Buchstaben.

Bei normalen Unicode-Schriften funktioniert die Suche und das Kopieren bei normalen westeuropäischen Texten ganz gut. Probleme bereiten aber manche der *OpenType Features*. Bei den Latin-Modern-Schriften kann man z B. mit

```
\setmainfont[Numbers=OldStyle]{Latin Modern Roman}
```

Zahlen in *oldstyle* setzen. Diese Zahlen können aber in der `pdf`-Datei weder gefunden noch kopiert werden!

Auch das «ß» verhält sich bei XeLaTeX anders: Mit pdfLaTeX wird sowohl «ß» (`\ss`) als auch die Großschreibung «SS» (`\SS`) als «ß» gefunden und kopiert. In XeLaTeX hingegen ist «SS» (`\SS`) von zwei S nicht zu unterscheiden.

Etwas Vergleichbares zum Paket `cmap`, mit dem man mit pdfLaTeX korrigierend eingreifen kann, wenn Suchen und Kopieren nicht funktionieren, gibt es mit XeLaTeX nicht.

Schriften laden mit `fontspec`

Das Paket `fontspec` [9] stellt ein geeignetes XeLaTeX-Benutzerinterface zur Verfügung, um die Schriften mit der neuen XeTeX-Methode zu laden.

Ich will hier nicht die Dokumentation von `fontspec` nacherzählen. Ich konzentriere mich lieber auf die wichtigsten Befehle und auf die Punkte, bei denen man vielleicht Probleme bekommt.

Die wichtigsten `fontspec`-Befehle

Das fast wichtigste Argument der folgenden Befehle ist das optionale Argument. Damit lässt sich die Schrift in vielen Details ändern und anpassen. Was theoretisch alles möglich ist, kann man der Dokumentation von `fontspec` entnehmen. Was praktisch bei den einzelnen Schriften möglich ist, muss man entweder ausprobieren, oder man schaut in dem (erweiterten) Eigenschaftsmenu der Schrift nach.

- `\fontspec[`⟨*Optionen*⟩`]{`⟨*Schriftname*⟩`}`

  Das ist der zentrale Befehl von `fontspec`. Mit ihm wechselt man die aktuelle Schrift*familie*, d. h. auch die Schriften, die man nach `\itshape` oder `\bfseries` erhält, ändern sich. `fontspec` wird sich alle Mühe geben, auch für die Varianten die «richtigen» Schriften zu finden. Ob das gelingt, hängt natürlich von der Schrift ab. Es ist aber problemlos möglich, über die Optionen die Auswahl anzupassen. Das Argument ist ein Schriftname in der Form, wie sie oben beschrieben wurde. `\fontspec` entspricht etwa einem `\fontfamily{`⟨*Familie*⟩`}\selectfont` in klassischem LaTeX.

  Der Befehl `\fontspec` wird in der Dokumentation des Paketes zwar viel benutzt, ist aber natürlich eigentlich ein interner Befehl, der in Dokumenten direkt nur in Ausnahmefällen verwendet werden sollte.

- `\setmainfont`, `\setsansfont`, `\setmonofont[`⟨*Optionen*⟩`]{`⟨*Schriftname*⟩`}`

  Mit diesen drei Befehlen kann man die Schriften für die drei üblichen Familien Roman (`\rmfamily`), Sansserif (`\sffamily`) und Typewriter (`\ttfamily`) festlegen. Die Wirkung der Befehle entspricht etwa den in diversen Schriftpaketen benutzten `\renewcommand\rmdefault{...}`.

- `\newfontfamily`⟨`\`*Befehl*⟩`[`⟨*Optionen*⟩`]{`⟨*Schriftname*⟩`}`

  Mit diesem Befehl kann man eine neue Schriftfamilie deklarieren, die dann analog zu `\rmfamily` über den *Befehl* ausgewählt werden kann. Ein entsprechender Textbefehl mit Argument wird aber nicht erzeugt. Das muss man schon selber tun:

```
\newfontfamily\verbfamily{Courier}
\DeclareTextFontCommand\textverb{\verbfamily}
```

- `\newfontface`⟨`\`*Befehl*⟩`[`⟨*Optionen*⟩`]{`⟨*Schriftname*⟩`}`

  Mit diesem Befehl kann man eine spezifische Schrift festlegen. `\`*Befehl* wirkt dann ähnlich wie ein `\usefont`-Befehl in LaTeX.

- `\defaultfontfeatures{`⟨*Optionen*⟩`}`

  Bei all den obigen Befehlen kann man über das optionale Argument viele einzelne Aspekte der Schrift oder Schriftfamilie ziemlich genau steuern, z. B. welche Ligaturen und welche Ziffern benutzt werden sollen. Mit `\defaultfontfeatures` kann man eine Liste solcher *Features* festlegen, die für alle nachfolgenden Schriften gilt. Die Liste gilt bis zum nächsten `\defaultfontfeatures` oder bis zum Ende der Gruppe.

- `\addfontfeatures`

  Mit diesem Befehl kann man *Features* zur aktuell im Text benutzten Familie hinzufügen, also lokal beispielsweise die Ziffern auf `\oldstyle` ändern. Die Wirkung endet beim Ende der aktuellen Gruppe oder wenn man die Familie wechselt, also z. B. nach Befehlen wie `\sffamily`, `\fontspec`, aber auch wenn man eine mit `\newfontface` definierte Schrift aufruft.

Die Defaultschriften von `fontspec`

Als Default lädt `fontspec` die Latin-Modern-Schriften, und zwar die OpenType-Versionen. Kurioserweise lädt `fontspec` die Schriften aber nicht mit seinen

eigenen Befehlen, sondern über das `euenc`-Paket und mit `fd`-Dateien. Das hat zur Folge, dass `\addfontfeatures` und einige Befehle aus dem `xltxtra`-Paket bei den Schriften gar nicht oder nur teilweise funktionieren.

Es kann daher sinnvoll sein, die Schriften nochmal zu laden:

```
\setmainfont[SmallCapsFont={* Caps},
          Mapping=tex-text]{Latin Modern Roman}
\setsansfont[Mapping=tex-text]{Latin Modern Sans}
\setmonofont[SmallCapsFont={* Caps}]{Latin Modern Mono}
```

Dies deklariert aber nicht alle Schriftvarianten, die man über das `euenc`-Paket bekommt. Es fehlen zum Beispiel die Familien `lmdunh`, `lmssq` und `lmvtt` und bei einigen Schriften die `sl`, `cond`, `demicond` und `lt`-Varianten. Natürlich kann man bei Bedarf auch diese Schriftformen nachdeklarieren und nutzen.

Kein `\slshape`

Der Autor von `fontspec` ist der Meinung, dass niemand neben einer *italic*-Schrift noch eine *slanted*-Variante braucht. Daher erhält man mit `\slshape` die gleiche Schrift wie mit `\itshape` – wenn man nicht gerade die `fontspec`-Version nutzt, bei der wegen eines Fehlers `\slshape` gar nicht geht. Ganz konsequent ist er aber dabei nicht: Das `euenc`-Paket benutzt und deklariert die *slanted*-Versionen der Latin-Modern-Schriften für die Roman- und die Mono-Familie. Lädt man aber, wie im vorherigen Abschnitt empfohlen, die Latin-Modern-Schriften über das `fontenc`-Interface nach, gehen, wie bereits erwähnt, diese Deklarationen verloren. Aber man kann sie mit dem folgenden Code nachholen:

```
\DeclareFontShape {EU1}{LatinModernRoman(0)}{m}{sl}
  {<->"Latin\space Modern\space Roman\space Slanted"}{}
\DeclareFontShape {EU1}{LatinModernMono(0)}{m}{sl}
  {<->"Latin\space Modern\space Mono\space Slanted"}{}
```

Mathematik

Die Mathematik-Schriften ändert `fontspec` nur wenig. Wenn es erkennt, dass ein Paket für Mathematikschriften, wie z. B. `fourier`, geladen wurde oder wenn die Paketoption `no-math` benutzt wurde, dann tut es (fast) gar nichts. In anderen Fällen setzt es die Schriftbefehle wie `\mathrm` und `\mathbf` auf

die Textschriften, die restlichen Mathematiksymbole und -schriften bleiben mehr oder weniger wie gehabt. Was genau angepasst wird, hängt u. a. von eventuell geladenen Paketen ab. Man sollte unbedingt anfangs sorgsam prüfen, ob Symbole und Schriften so sind, wie gewünscht.

Bei Mathematikschriften benötigt TeX weit mehr `\fontdimen`-Parameter als bei Textschriften, um die Hoch- und Tiefstellung der vielen Indices korrekt zu berechnen. XeTeX kann im Prinzip diese Parameter aus OpenType-Mathematikschriften auslesen (und daher solche Schriften auch nutzen). Was aber noch fehlt, ist das Äquivalent zu `xunicode`: Für all die in Formeln benutzten Befehle wie `\int` oder `\alpha` muss es ja sinnvolle Unicode-Definitionen geben. Ein experimentelles Paket dazu ist `unicode-math` [11].

Literatur

[1] François Charette: *Polyglossia Package*; CTAN: `macros/xetex/latex/polyglossia/`.

[2] *Font properties extension*; `http://www.microsoft.com/typography/TrueTypeProperty21.mspx`.

[3] Michel Goossens: *The XeTeX Companion*; 2008; `http://xml.web.cern.ch/XML/lgc2/xetexmain.pdf`.

[4] Michel Goossens, Frank Mittelbach et al.: *The LaTeX Graphics Companion 2nd ed*; Addison-Wesley Publishing Company; Reading, Mass.; 2007.

[5] Jonathan Kew: *XeTeX-PSTricks*; CTAN: `graphics/xetex-pstricks/`.

[6] Jonathan Kew: *XeTeX*; 2008; `http://scripts.sil.org/XeTeX`; `https://sourceforge.net/projects/xetex/`.

[7] *The LaTeX Graphics Companion Supplementary Material*; 2008; `http://xml.web.cern.ch/XML/lgc2/`.

[8] Ross Moore: *Xunicode Package*; CTAN: `macros/xetex/latex/xunicode/`.

[9] Will Robertson: *Fontspec Package*; CTAN: `macros/xetex/latex/fontspec/`.

[10] Will Robertson: *Ifxetex Package*; CTAN: `macros/generic/ifxetex/`.

[11] Will Robertson: *Unicode-math Package*; XeTeX-SVN: `texmf/source/xelatex/unicode-math/`.

[12] Will Robertson: *Xltxtra Package*; CTAN: `macros/xetex/latex/xltxtra/`.

[13] *XeTeX-Mailingliste*; `http://tug.org/mailman/listinfo/xetex`.

[14] *XeTeX-SVN*; `http://scripts.sil.org/svn-public/xetex/TRUNK/`.

Dokumentenmanagement mit LaTeX und Subversion

Uwe Ziegenhagen

Die Nutzung eines Versionsverwaltungssystems bietet eine Reihe von Vorteilen in der täglichen Arbeit von Programmierern und Autoren. Die Zusammenarbeit in einem Team vereinfacht sich drastisch, da der mühsame und fehlerträchtige Austausch von Dateien über FTP oder E-Mail entfällt, ältere Versionen einer Datei können problemlos wiederhergestellt werden und die Zusammenführung verschiedener Versionen vereinfacht sich. Ein nicht zu unterschätzender Vorteil ist zudem die Möglichkeit, Backups quasi im »Vorbeigehen« zu erstellen.

Mit Subversion existiert eine moderne Versionsverwaltung, die auf allen gängigen Plattformen genutzt werden kann und die wenig Konfigurations- und Einarbeitungszeit erfordert. In diesem Artikel geht es um die Nutzung von Subversion mit LaTeX. Erläutert wird die Installation und Konfiguration auf Windows und Linux-Systemen, außerdem werden einige Pakete besprochen, die eine bequeme Integration von Subversion-Information in LaTeX ermöglichen.

CVS und Subversion

Im Unterschied zum weit verbreiteten *Concurrent Versions System* (CVS) bezieht sich die Versionisierung von Subversion nicht auf einzelne Dateien, sondern auf ein ganzes Verzeichnis oder einen Verzeichnisbaum. Jede Versionsnummer n bezieht sich auf den Zustand dieses Verzeichnisses, nachfolgend Repository genannt, nach der n-ten Übermittlung der Daten, dem sogenannten *Commit*. Wenn also über eine Datei in Version 4 gesprochen wird, bezieht man sich auf die Datei, so wie sie in der vierten Revision existierte.

Bei jedem Check-out von Dateien aus einem Subversion-Repository wird diejenige Version ausgeliefert, deren Version kleiner oder gleich der gewünschten Versionsnummer ist. Kopien der ausgelieferten Dateien speichert Subversion in einem speziellen Verzeichnis (`.svn`) bei jedem Check-out, Update und Commit. Obwohl dies den benötigten Festplattenplatz verdoppelt, hat es doch einen entscheidenden Vorteil: Änderungen lassen sich auch ohne Netzwerkzugang

nachverfolgen und beim Commit muss Subversion nicht die gesamte Datei, sondern nur die jeweiligen Unterschiede zum Server übermitteln. CVS im Gegensatz muss die ganze Datei übermitteln, da die Änderungen serverseitig ermittelt werden. Subversion-Commits sind atomar, d. h. Änderungen an einer Datei werden entweder komplett oder überhaupt nicht gespeichert. Eventuelle Netzwerk-Probleme können also nicht zu inkonsisten Repositories führen.

Installation

Für die Installation von Subversion gibt es verschiedene Möglichkeiten und Vorgehensweisen. Man kann entweder Svnserve [5] als eigenständigen Dienst bzw. Daemon nutzen oder aber Subversion als Apache-2-Modul installieren, das WebDAV[1], eine Erweiterung des HTTP-Protokolls nutzt.

In diesem Artikel werde ich auf beide Installationsarten unter Windows und Ubuntu Linux eingehen. Der eigenständige Dienst lässt sich zwar einfacher installieren, bei der Nutzung von Subversion als Apache-Modul gibt es aber zwei – je nach Situation nützliche – Aspekte: Man kann auf ein Repository mit dem Web-Browser zugreifen, außerdem können die in Apache eingebauten Möglichkeiten der Nutzer-Authentifizierung genutzt werden.

Windows XP

Installation als Apache-Modul

Binär-Versionen von Apache 2 sind verfügbar von [1]; persönlich bevorzuge ich aber die Nutzung einer WAMP-Umgebung[2] wie beispielsweise [7]. Wir entpacken xampp.zip[3] nach `C:/xampp` und starten den Apache-Server über die `xampp-control.exe`. Wenn wir nun die Webseite `http://localhost` in einem Browser öffnen, sollte die Startseite, wie in Bild 1 dargestellt, angezeigt werden.

Bei einer lokalen Installation sollte man entsprechende Sicherheitsmaßnahmen gegen den Zugriff aus dem Internet ergreifen. Entsprechende Informationen findet man unter dem Punkt `security` im xampp-Menü sowie in der Apache-Dokumentation [11].

Wir laden die Binär-Version von Subversion von [4] herunter und extrahieren alle Dateien aus dem Zip-Archiv nach `C:/Programme/Subversion`. Nachdem

[1] *Web-based Distributed Authoring and Versioning*
[2] Windows-Apache-MySQL-PHP
[3] Aktuelle Version: 1.6.7

Abbildung 1: Screenshot der xampp Startseite

wir den Pfad zum Verzeichnis `C:/Programme/Subversion/bin` aus der Umgebungsvariablen PATH von Windows hinzugefügt haben, können wir `svn help` aus der Kommandozeile starten, um die Installation zu prüfen. Im nächsten Schritt kopieren wir `mod_authz_svn.so` und `mod_dav_svn.so` aus dem Verzeichnis `subversion/bin` in das Apache-Verzeichnis `modules` und überschreiben eventuell vorhandene ältere Versionen.

Im letzten Schritt schalten wir im Apache die WebDAV-Erweiterung und das Subversion-Modul ein, indem wir folgende Zeilen der `httpd.conf` im Apache-Verzeichnis `/conf` hinzufügen:

- `LoadModule dav_svn_module modules/mod_dav_svn.so` und
- `LoadModule authz_svn_module modules/mod_authz_svn.so`

Bevor wir Apache neu starten, ist der allerletzte Schritt die Erstellung des Stamm-Verzeichnisses für alle Repositories `c:/allMyRepositories` und die Ergänzung der `httpd.conf` um Listing 1:

Listing 1: Setup code for the Windows repository root

```
<Location /svn>
DAV svn

SVNParentPath c:/allMyRepositories
</Location>
```

Mit der Kommandozeile wechseln wir nach `c:/allMyRepositories` und erstellen unser erstes Repository mittels `svnadmin create test`.

Wenn wir jetzt `http://localhost/svn/firstsample/` im Browser öffnen, sollten wir ein leeres Verzeichnis-Listing mit der Überschrift `Revision 0: /` sehen. Die grundlegende Subversion-Installation ist damit abgeschlossen, durch die Installation eines grafischen Subversion-Clients wie TortoiseSVN lässt sich aber die Arbeit mit Subversion deutlich erleichtern.

TortoiseSVN [6] ist ein freier Subversion-Client für Windows, der seine Funktionen über das Menü der rechten Maustaste zugänglich macht.

Das TortoiseSVN-Interface fügt sich komplett in den Windows Explorer ein; Icons zeigen in einem Arbeitsverzeichnis sofort an, welche Dateien und Verzeichnisse verändert wurden und in das Repository übermittelt werden müssen. Die Installation ist einfach, nach dem Neustart des Systems finden wir im Kontextmenü die entsprechenden Menüeinträge. Neben TortoiseSVN gibt es noch eine Reihe weiterer Clients, zum Beispiel RapidSVN (Windows, Unix/Linux) und SVNcommander (Linux), Entwicklungsumgebungen wie Netbeans und Eclipse bieten auch eingebauten Subversion-Support.

Installation als Dienst

Die Installation von `svnserve` unter Windows ist in wenigen Schritten erledigt. Wir laden den Subversion-Installer von `http://subversion.tigris.org` herunter und führen die Installation durch. Anschließend legen wir ein `Repository`-Verzeichnis an (hier `c:/repos`) und erstellen über die Kommandozeile – oder gegebenenfalls TortoiseSVN – ein erstes Repository: `svnadmin create test` und ändern die Lese- und Schreibrechte in der Datei `conf/svnserve.conf` gemäß Listing 5, um eventuelle Fehler leichter eingrenzen zu können. Über die Kommandozeile starten wir dann auch den svnserve-Dienst mittels `svnserve.exe -d -r c:/repos`. Per TortoiseSVN oder Kommandozeile (`svn co svn://localhost/test` sollte ein Check-out jetzt möglich sein.

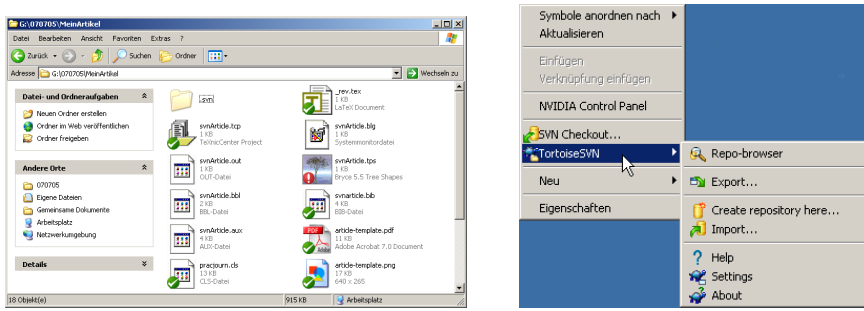


Abbildung 2: Screenshot eines Arbeitsverzeichnisses mit TortoiseSVN Kontextmenü von Subversion 1.4.4.

Der Nachteil dieser Lösung ist klar erkennbar; sobald das Kommandozeilenfenster geschlossen wird, ist auch der Subversion-Server nicht mehr verfügbar. Daher werden wir im nächsten Schritt `svnserve` als Windows-Dienst konfigurieren. Dazu führen wir den Befehl in Listing 2 aus, die Eingabe muss in einer Zeile erfolgen. Zum Abschluss kann der Dienst über das Dienste-Menü in der Windows-Verwaltung gestartet werden.

Listing 2: Befehl zur Installation als Windows-Dienst

```
sc create svnserve binpath="C:\Programme\Subversion\bin\svnserve.exe
--service --root c:/repos" displayname= "Subversion"
depend= tcpip start= auto
```

Linux (Ubuntu 8)

Installation als Apache-Modul

Die Installation unter Ubuntu 8 ist ebenso leicht wie die Installation unter Windows. Mit `sudo apt-get install` oder dem Paketmanager Synaptic installieren wir die folgenden Pakete:

- apache2,
- libapache2-svn,
- subversion.

Weitere eventuell notwendige Pakete werden durch das Paketmanagement selbstständig ausgewählt und konfiguriert. Nach der Installation sind die letzten Schritte die Erstellung eines Stammverzeichnisses für unsere Repositories, hier beispielsweise in `/home/uwe/repositoryRoot`, die entsprechende Anpassung von `/etc/apache2/sites-available/default` und das Setzen der notwendigen Rechte mittels `chmod -R 777 /home/uwe/repositoryRoot`:

Listing 3: Anpassung für die Datei `/etc/apache2/sites-available/default`

```
<Location /svn>
DAV svn

SVNParentPath /home/uwe/repositoryRoot
</Location>
```

In `/home/uwe/repositoryRoot` erstellen wir dann per `svnadmin create test` ein erstes Repository, das unter der URL `http://localhost/svn/test` im Browser angezeigt werden sollte. Gibt es bei diesem Schritt einen Zugriffsfehler, sind vermutlich die Rechte nicht richtig gesetzt worden.

Erste Schritte

Um unser Repository zu füllen, erstellen wir in einem Verzeichnis (alle enthaltenen Dateien werden gleich importiert) ein kleines LaTeX-Dokument (articletemplate.tex):

Listing 4: Eine einfache LaTeX-Datei

```
\documentclass{article}
\begin{document}
  Hello World!
\end{document}
```

Mittels Kommandozeile (`svn import http://localhost/svn/firstsample/ -m 'Anfangsimport'`) oder dem entsprechenden Eintrag im TortoiseSVN-Kontextmenü können wir jetzt die Datei importieren, `'Anfangsimport'` ist der optionale Parameter für Kommentar beim Import.

Apache zeigt jetzt `Revision 1: /` im Browser (siehe Bild 3). Um jetzt an dieser Datei Änderungen vornehmen zu können, müssen wir sie in ein `Arbeitsverzeichnis` auschecken. Alle weiteren Commits werden aus diesem Verzeichnis ausgeführt.

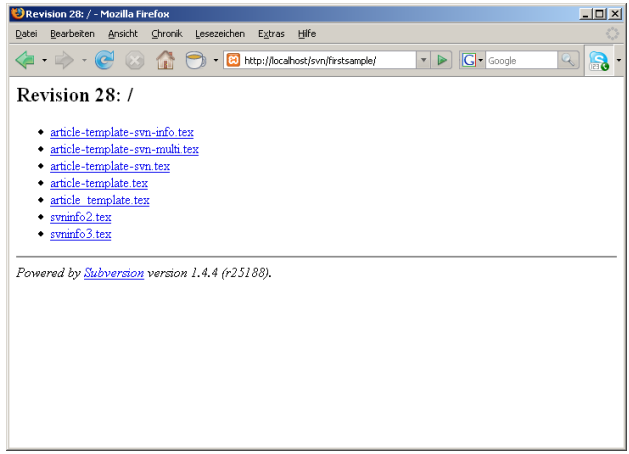


Abbildung 3: Repository mit Apache und dem Subversion-Modul

Installation als Dienst

Die Installation von `svnserve` ist noch deutlich einfacher als die Installation des Apache-Moduls. Mittels `sudo apt-get install subversion` wird die Subversion-Installation gestartet, Ubuntu lädt und installiert dann alle benötigten Pakete. Aus Sicherheitsgründen ist es ratsam, den `svnserve`-Dämon nicht unter dem `root`-Account laufen zu lassen. Wir werden daher einen Nutzer `svn` anlegen, der in seinem Home-Verzeichnis die Repository-Verzeichnisse enthält:

- `sudo useradd svn` legt den Nutzer `svn` an,
- `sudo passwd svn` setzt Passwort für den Nutzer,
- `sudo mkdir /home/svn` erstellt Home-Verzeichnis,
- `sudo chown -R svn /home/svn` setzt die korrekten Besitzrechte.

Nachdem die entsprechenden Verzeichnisse und Rechte gesetzt sind, kann jetzt das erste Repository angelegt werden. Wir wechseln per `su - svn` zum Nutzer `svn` und erstellen in dessen Homeverzeichnis das Stammverzeichnis, das die Repositories enthalten wird: `mkdir repos`. In diesem Verzeichnis nut-

zen wir `svnadmin create test`, um die von Subversion genutzten Datei- und Ordnerstrukturen anzulegen.

Für das erste Beispiel ist es sinnvoll, die Rechte bewusst so zu setzen, dass jedermann Lese- und Schreibrechte auf dieses Repository hat. Eventuelle Probleme lassen sich so leichter erkennen. Dazu bearbeiten wir die Datei `svnserve.conf`, die in `test/conf` liegt, gemäß Listing 5. In der späteren Arbeit mit Subversion, insbesondere wenn mehrere Autoren an einem Projekt beteiligt sind, ist das Anlegen von Nutzerkonten ratsam.

Listing 5: geänderter Abschnitt aus `svnserve.conf`

```
[general]
### These options control access to the repository for unauthenticated
### and authenticated users.  Valid values are "write", "read",
### and "none".  The sample settings below are the defaults.
anon-access = write
auth-access = write
```

Im letzten Schritt starten wir den Dienst mit `svnserve -d -r /home/svn/repos` und importieren eine beliebige Datei in unser Repository mittels `svn import -m 'initial import' svn://localhost/test`, die wir dann in ein beliebiges Verzeichnis auschecken können: `svn co svn://localhost/test`.

Ebenso wie unter Windows ist es ratsam, den Subversion-Dienst automatisch starten zu lassen oder die Kommunikation über SSH zu verschlüsseln, siehe dazu die Hinweise unter [5].

Integration mit LaTeX

Um Subversion in unseren LaTeX-Workflow zu integrieren, müssen wir in die TeX-Datei entsprechende Meta-Informationen einfügen. Die folgende Liste enthält die verfügbaren Schlüsselwörter und ihre Beschreibung:

Date (LastChangedAt) Datum und Zeit des letzten Checkin,
Revision: (LastChangedRevision) Revisionsnummer,
Author: (LastChangedBy) Name des Autors,
HeadURL: URL der Datei,
Id: Zusammenfassung der anderen Schlüsselwörter.

Wenn man mit der Kommandozeile in das Arbeitsverzeichnis wechselt und diese Schlüsselwörter mittels `svn propset svn:keywords 'Date HeadURL Revision Id' article_template.tex` zur Expansion freigibt, erweitert Subversion diese in der Datei `article_template.tex`, jeweils eingeschlossen in $-Zeichen. In TortoiseSVN lassen sich die Schlüsselwörter auch bequem über den Menü-Punkt `Properties` einfügen.

Listing 6: Die Beispieldatei mit expandierten Schlüsselwörtern

```
% $Revision: 10 $
% $HeadURL: http://tools.assembla.com/svn/svnArticle/svnArticle.tex $
% $Date: 2007-07-22 19:50:04 +0200 (So, 22 Jul 2007) $
% $Author$
% $Id: svnArticle.tex 10 2007-07-22 17:50:04Z uweziegenhagen $

\documentclass{article}
\begin{document}
 Hello World!
\end{document}
```

Alle LaTeX-Pakete, die ich im weiteren Artikel vorstellen werde, basieren auf der Auswertung dieser Schlüsselwörter.

svn

Das Paket `svn` gestattet den Zugriff auf die Subversion-Informationen über die Syntax `\SVN $Keyword: <metadata>$`. Wenn die Schlüsselwörter korrekt expandiert wurden, stehen die folgende LaTeX-Befehle zur Verfügung:

- `\SVNDate` Datum des letzten Commit, `\SVNTime` als Commit-Zeit und `\SVNRawDate` als Datum und Zeit im Rohformat, wenn `Keyword` $Date$ war;
- `\SVNKeyword` sonst (Beispiele: \SVNId, \SVNHeadURL)

Listing 7: Beispieldatei mit eingefügten `svn`-Befehlen

```
\documentclass{article}
\usepackage{svn}

\SVN $Id: svnArticle.tex 10 2007-07-22 17:50:04Z uweziegenhagen $
```

```
\SVN $Date: 2007-07-22 19:50:04 +0200 (So, 22 Jul 2007) $
\SVN $Id: svnArticle.tex 10 2007-07-22 17:50:04Z uweziegenhagen $
\SVN $HeadURL: http://tools.assembla.com/svn/svnArticle/svnArticle.tex $

\begin{document}

\SVNDate \\
\SVNRawDate \\
\SVNTime \\
\SVNId \\
\SVNHeadURL
\end{document}
```

July 15, 2007
2007-07-15 17:33:30 +0200 (So, 15 Jul 2007)
17:33:30
article-template.tex 12 2007-07-15 15:33:30Z
http://localhost/svn/firstSample/article-template.tex

Abbildung 4: Ausgabe von article-template.tex mit dem Paket `svn`

svninfo

Das Paket `svninfo` bezieht die Versionsinformationen ausschließlich aus dem Tag Id, das bei der Expansion von `\svnInfo` erzeugt wird: \svnInfo $Id: article-template-svn-info.tex 18 2007-07-15 16:11:21Z$

Um die SVN-Informationen nutzen zu können, werden die folgenden Kommandos definiert:

- `\svnInfoFile` Dateiname,
- `\svnInfoRevision` Revisionsnummer,
- `\svnInfoDate` Datum des letzten check-in,
- `\svnInfoTime` Zeit des letzten check-in,
- `\svnInfoYear` Jahr wie in \svnInfoDate,

- `\svnInfoMonth` Monat wie in \svnInfoDate,
- `\svnInfoDay` Tag wie \svnInfoDate,
- `\svnInfoOwner` Besitzer der Datei,
- `\svnToday` Datum des letzten Check-in im `\today` Format,
- `\svnInfoMinRevision` minimale Revision des gesamten Dokuments,
- `\svnInfoMaxRevision` maximale Revision des gesamten Dokuments.

`\svnInfoMinRevision` und `\svnInfoMaxRevision` sind nützlich für Dokumente, die aus mehreren Dateien bestehen. Das Paket bietet noch weitere Optionen wie `fancyhdr, eso-foot, scrpage2`, um Subversion-Informationen am Rand oder in der Fußzeile des Dokuments auszugeben. Details dazu finden sich in der Dokumentation des Pakets.

svn-multi

Das Paket `svnmulti` stellt zwei Kommandos bereit, `\svnid` und `\svnidlong`, um Subversion-Informationen zu verwerten. Zur Ausgabe im Dokument werden folgende Makros definiert:

- `\svnrev` Revisionsnummer,
- `\svndate` Datum des letzten Check-in,
- `\svnauthor` Autor,
- `\svnfilerev` Revision der aktuellen Datei, falls dieses einen `\svnid` oder `\svnidlong` Befehl enthält oder die entsprechenden Werte der letzten Datei, falls keiner dieser Befehle in der aktuellen Datei enthalten ist.
- `\svnmainurl` und `\svnmainfilename` geben die URL beziehungsweise den Namen der Hauptdatei aus, wenn diese am Ende der Dokumentenpräambel mit `\svnmainfile` festgelegt wurde.

Weiterhin nutzt das Paket `\svn{`*`Schlüssel`*`}` und `\svnkw{`*`Schlüssel`*`}`, um Subversion-Schlüsselwörter direkt auszugeben. Um auf Datumsinformationen zuzugreifen, stellt das Paket noch eine Reihe weiterer Makros bereit (die Bedeutung erschließt sich aus dem Namen): `\svnfileyear`, `\svnfilemonth`, `\svnfileday`, `\svnfilehour`, `\svnfileminute`, `\svnfilesecond`, `\svnfiletimezone`, `\svnyear`, `\svnmonth`, `\svnday`, `\svnhour`, `\svnminute`, `\svnsecond` und `\svntimezone`.

Zusammenfassung

Dieser Artikel beschreibt die Installation von Subversion unter Linux und Windows und stellt drei Pakete vor, um Subversion-Informationen in LaTeX-

Dokumente zu integrieren. Mehr Informationen findet sich in den Dokumentationen der einzelnen Pakete sowie in der Literatur zu Subversion ([10, 13]. Ein Feedback zu diesem Artikel, ist jederzeit willkommen, Aktualisierungen und Errata werde ich unter `http://www.uweziegenhagen.de/latex/` verfügbar machen.

Literatur

[1] *Apache 2 web server*; http://httpd.apache.org.

[2] *psvn*; `http://www.xsteve.at/prg/vc_svn/`.

[3] *RapidSVN*; `http://rapidsvn.tigris.org`.

[4] *Subversion (Software)*; `http://subversion.tigris.org/`.

[5] *Svnserve Based Server*; `http://tortoisesvn.net/docs/nightly/TortoiseSVN_en/tsvn-serversetup-svnserve.html`.

[6] *TortoiseSVN*; `http://tortoisesvn.tigris.org`.

[7] apachefriends.org: *Xampp*; `http://www.apachefriends.org/`.

[8] Paul A. Blaga: *PracTeX Journal: Making an Electronic Journal with web tools, Wiki, and version control*; *The PracTeX Journal*; 2; 2007; `http://tug.org/pracjourn/2007-2/blaga/`.

[9] Achim D. Brucker: *svninfo*; `http://www.ctan.org/tex-archive/macros/latex/contrib/svninfo/`.

[10] Ben Collins-Sussman, Brian W. Fitzpatrick und C. M. Pilato: *Version Control with Subversion. Next Generation Open Source Version Control*; O'Reilly; 2004.

[11] Apache Foundation: *Apache HTTP Server Version 2.2 Documentation*; `http://httpd.apache.org/docs/2.0/en/`.

[12] Richard Lewis: *svn*; `http://www.ctan.org/tex-archive/macros/latex/contrib/svn/`.

[13] Mike Mason: *Pragmatic Version Control Using Subversion*; Pragmatic Programmers LLC.; 2006.

[14] Martin Scharrer: *svn-multi*; vormals svnkw.

[15] wikipedia.org: *Subversion_(software)*; in German.

Das »große scharfe S« wurde normiert

Markus Kohm

Am Donnerstag, den 26. Juni 2008 war in d.c.t.t. zu lesen: »Soeben wurde das ›große scharfe S‹ normiert.«

Was lediglich bedeutet, dass es ein solches gibt und es eine Unicode-Position hat. Damit sind einige technischen Voraussetzungen geschaffen, um ein solches Zeichen darstellen zu können. Die Regeln der Rechtschreibung sehen die Verwendung aber deshalb noch lange nicht vor und werden das in den nächsten Jahren auch nicht. Solange das Zeichen nicht einfach per Shift-ß eingegeben werden kann, wird es auch kaum eine Abstimmung mit den Fingern in Richtung zur Verwendung des neuen Zeichens geben. Man wird also auch zukünftig wesentlich häufiger ein falsch verwendetes kleines ß im Versalsatz finden als ein großes.

Die Herausforderung für LaTeX ist im übrigen nicht sonderlich groß. LaTeX kennt im T1-Encoding nämlich schon lange einen »upcase« für ß:

```
\documentclass{article}
\usepackage[T1]{fontenc}
\usepackage[ngerman]{babel}
\begin{document}
\pagestyle{empty}
\MakeUppercase{"s} oder \SS\ im Vergleich zu \ss.
\end{document}
```

Dass die Ausgabe derzeit wie SS aussieht, ist nebensächlich. Bei Zeichensätzen, die ein Versal-ß enthalten, kann man diese Position (optional) sehr gut damit belegen.

Über gute Formen für Versal-ß diskutieren Typografen schon lange. Es gibt da einige nette Vorschläge. Die Herausforderungen sind für das Zeichen nicht wirklich größer als für andere Zeichen auch.

Von fremden Bühnen

Neue Pakete auf CTAN

Jürgen Fenn

Der Beitrag stellt neue Pakete auf CTAN seit April 2008 bis zum Redaktionsschluss vor. Die Liste folgt der umgekehrten chronologischen Reihenfolge. Bloße Updates werden nicht aufgeführt. Sie können auf der moderierten *tex-announce*-Mailingliste verfolgt werden, die auch unter `http://blog.gmane.org/gmane.comp.tex.ctan.announce` online verfügbar ist.

`lnotes` von *Alpha Huang* ist eine neue chinesische Einführung in (LA)TEX.
`CTAN:info/lnotes`

`latex-course` von *Engelbert Buxbaum* ist ein kleiner LATEX-Kurs in Form einer `beamer`-Präsentation in englischer Sprache.
`CTAN:info/latex-course`

`lshort-chinese` von *Zhaopeng Xing* ist die chinesische Übersetzung von `lshort-german`.
`CTAN:info/lshort/chinese`

`polyglossia` von *François Charette* ersetzt `babel` bei Verwendung von XETEX. 54 Sprachen werden unterstützt.
`CTAN:macros/xetex/latex/polyglossia`

`xecjk` von *Wenchang Sun* ist ein CJK-Support für XETEX.
`CTAN:macros/xetex/latex/xecjk`

`inlinedef` von *Stephen Hicks* ist eine Erweiterung zur Schachtelung von Definitionen mit `\def` oder `\gdef`.
`CTAN:macros/latex/contrib/inlinedef`

`scrindex` aus dem `oberdiek`-Bundle von *Heiko Oberdiek* definiert die Umgebung `theindex` aus dem Paket `index` neu, wenn eine Klasse aus dem KOMA-Script-Bundle geladen wird. Außerdem wird die KOMA-Script-Option `idxtotopic` unterstützt.
`CTAN:macros/latex/contrib/oberdiek`

`tdsfrmath` von *Yvon Henel* ist eine Makro-Sammlung für Mathematiklehrer an französischen Gymnasien.
`CTAN:macros/latex/contrib/tdsfrmath`

`xepersian` von *Vafa Khalighi* dient zum Setzen persischer und arabischer Texte mit XETEX.
`CTAN:macros/xetex/latex/xepersian`

`xstring` von *Christian Tellechea* dient zum Bearbeiten und von Strings (Tests auf bestimmte Zeichenfol-

gen; Extraktion und Ersetzen von Substrings; diverse Berechnungen).
CTAN:macros/latex/contrib/xstring

`biblatex-chem` von *Joseph Wright* sind experimentelle `biblatex`-Stile zur Verwendung in der Chemie. Unterstützt werden Zeitschriften von ACS, RSC- und Wiley.
CTAN:macros/latex/exptl/biblatex-contrib/biblatex-chem

`eukdate` von *Andrew Gilbert Moschou* gibt das Datum einschließlich Wochentag mittels `\today` im britischen Format aus, z. B.: *»Saturday, 26 June 2008«*.
CTAN:macros/latex/contrib/eukdate

`pgfopts` von *Joseph Wright* setzt auf dem Paket `pgfkeys` auf und ermöglicht es, Daten, die in einem *key=vaule*-Interface vorliegen, als Optionen in eine LaTeX-Klasse zu nutzen. Es entspricht damit den Paketen `kvoptions` und `keyval`.
CTAN:macros/latex/contrib/pgfopts

`elsarticle` von *Simon Pepping* ist eine neue LaTeX-Klasse für Artikel des Elsevier-Verlags.
CTAN:macros/latex/contrib/elsarticle

`vwcol` von *Will Robertson* erlaubt das Setzen von mehrspaltigen Absätzen mit unterschiedlicher Breite auf einer Seite.
CTAN:macros/latex/contrib/vwcol

`blowup` von *Rolf Niepraschk* dient zum Hoch- oder Herunterskalieren aller Seiten eines Dokuments. Es ähnelt der TeX-Primitive `\magnification`, ist aber genauer und benutzerfreundlicher.
CTAN:macros/latex/contrib/blowup

`forarray` von *Christian Schröppel* dient zum Setzen von Listen und Matrizen. TeX- und LaTeX-Befehle können innerhalb von Matrizen verwendet werden, die auch geschachtelt werden können.
CTAN:macros/latex/contrib/forarray

`encxvlna` von *Zdenek Wagner* erzeugt geschützte Leerzeichen nach bestimmten, kurzen Präpositionen und Konjunktionen in tschechischen und slovakischen Texten.
CTAN:macros/generic/encxvlna

`ean13isbn` von *Zdenek Wagner* erzeugt EAN13-Strichcode für ISBN-Buchnummern.
CTAN:macros/latex/contrib/ean13isbn

`zwgetfdate` von *Zdenek Wagner* liest die Versionsdaten aus Paketen und Dateien, die in ein Dokument eingebunden werden, und stellt sie zur weiteren Verwendung in Makros bereit, was vor allem praktisch ist, wenn man nicht `doc`/`docstip` verwendet.
CTAN:macros/latex/contrib/zwgetfdate

`hyph-utf8` von *Jonathan Kew*, *Mojca Miklavec* und *Arthur Reutenauer* sind neue Trennmuster in UTF-8-Kodierung, die die alten LaTeX-Trennmuster ersetzen sollen.
CTAN:language/hyph-utf8

`dehyph-exptl` von *Stephan Hennig* sind neue, experimentelle deutsche Trennmuster für die alte und die neue deutsche Rechtschreibung. Sie können mit den Paketen `babel` und `hyphsubst` aus dem `oberdiek`-Bundle verwendet werden.
CTAN:language/hyphenation/dehyph-exptl

`xetex-pstricks` von *Jonathan Kew* enthält Konfigurationsdateien zur Verwendung von PSTricks mit XeTeX und `xdvipdfmx`.
`CTAN:graphics/xetex-pstricks`

`tikz-inet` von *Marc de Falco* erweitert `tikz` um Makros zum Zeichnen von *interaction nets*.
`CTAN:graphics/pgf/contrib/tikz-inet`

`xetexfontinfo` von *Jonathan Kew* enthält Musterdokumente, die zur Demonstration der verfügbaren Features von Fonts mit XeTeX.
`CTAN:macros/xetex/plain/xetexfontinfo`

`hyphsubst` von *Heiko Oberdiek* dient zum Ersetzen von Trennmustern durch andere Muster, die im TeX-Formatfile angegeben sind.
`CTAN:macros/latex/contrib/oberdiek`

`letltxmacro` von *Heiko Oberdiek* erlaubt es, den TeX-Befehl `\let` auch in LaTeX-Befehlen zu verwenden, die mit `\DeclareRobustCommand` definiert worden sind oder die optionale Argumente haben.
`CTAN:macros/latex/contrib/oberdiek`

`fontwrap` von *Mike »Pomax« Kamermans* dient zum schnellen Umschalten zwischen Unicode-Fontendcofdings in mehrsprachigen Texten.
`CTAN:macros/xetex/latex/fontwrap`

`tufte-latex` von *Kevin Godby* enthält LaTeX-Klassen, zu deren Gestaltung der Autor von den Büchern und Handouts des Informatikers und Grafikdesigners *Edward Tuftes* inspiriert wurde.
`CTAN:macros/latex/contrib/tufte-latex`

`imtekda` von *Simon Dreher* ist die auf KOMA-Script aufbauende Klasse zum Setzen von wissenschaftlichen Arbeiten am *Institut für Mikrosystemtechnik (IMTEK)* an der Universität Freiburg.
`CTAN:latex/macros/contrib/imtekda`

`dviasm` von *Jin-Hwan Cho* ist ein Skript zum Editieren von DVI-Dateien (vgl. `http://tug.org/TUGboat/Articles/tb28-2/tb89cho.pdf`).
`CTAN:dviware/dviasm`

`javadoc` von *Jolle Kiesel* dient zum Dokumentieren (nicht nur) von Java-Quelltext.
`CTAN:macros/latex/contrib/javadoc`

`theoremref` von *Emil Jerabek* dient zum Einfügen von Querverweisen auf mathematische Sätze.
`CTAN:macros/latex/contrib/theoremref`

`mpman-ru` von *Vladimir Lidovski* ist die russische Übersetzung des METAPOST-Manuals.
`CTAN:info/metapost/doc/russian/mpman-ru`

`ctantools` von *Kyanh* ist ein Unix-Skript, mit dem man in der Ausgabe von `http://ctan.org/tex-archive/macros/latex/contrib/` LaTeX-Pakete suchen kann. `wget`, `lynx`, `gawk`, `wc` und `sed` werden hierzu benötigt.
`CTAN:support/ctantools`

`mathtype` von *Design Science, Inc.* ist ein mathematischer Formeleditor für Microsoft Windows (30-Tage Testversion).
`CTAN:support/mathtype`

`stex` von *Michael Kohlhase* erlaubt es, TeX- und LaTeX-Dokumente nach den Regeln des *mathematical*

knowledge management (MKM) semantisch auszuzeichnen.
CTAN:macros/latex/contrib/stex

`cyklop` von *Janusz Marian Nowacki* ist ein Font, der in den 20er Jahren entworfen wurde. Er steht im Type1- und im OpenType-Format zur Verfügung.
CTAN:fonts/cyklop

`lshort-slovenian` von *Bor Plestenjak* ist die slovenische Übersetzung von *l2kurz*.
CTAN:info/lshort/slovenian

`lcyw` von *Vladimir Lidovski* ist die LaTeX-Unterstützung für die klassischen kyrillischen CM-Fonts.
CTAN:macros/latex/contrib/lcyw

`miktex_update` von *Josef Kleber* ist ein Bash-Skript für Cygwin zum automatischen Update von MiKTeX, mit dem auch neue Pakete installiert werden können.
CTAN:support/miktex_update

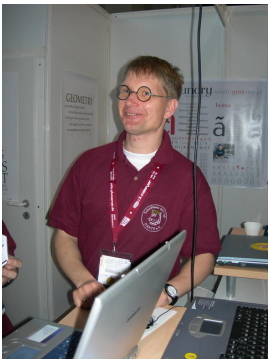
`isonums` von *Luis Rivera* baut auf `ziffer` auf und stellt Zahlen im Mathematikmodus gemäß ISO 31-0 dar, egal in welchem Format sie eingegeben worden waren.
CTAN:macros/latex/contrib/misc/isonums.sty

`edmargin` von *John Burt* bietet Erweiterungen zum Setzen von Endnoten in textkritischen Ausgaben.
CTAN:macros/latex/contrib/edmargin

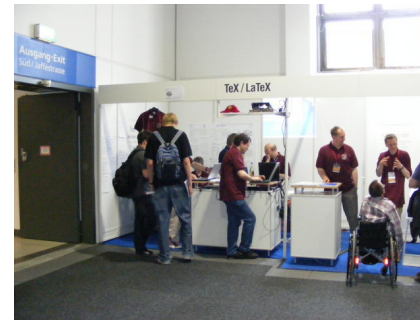
`siunitx` von *Joseph Wright* will die Features aller bisherigen LaTeX-Pakete zum Setzen von Werten mit den dazugehörigen Einheiten in einem Paket zusammenfassen.
CTAN:macros/latex/exptl/siunitx

`varsfromjobname` von *Uwe Ziegenhagen* erlaubt es, mit Befehlen wie `\getonefromjobname` auf die mit Bindestrichen abgetrennten Teile des Namens der `tex`-Datei zuzugreifen.
CTAN:macros/latex/contrib/varsfromjobname

Linuxtag in Berlin



(Fotos: Meike Schmedt und Karlheinz Geyer)



(Fotos: Meike Schmedt und Karlheinz Geyer)

Spielplan

Termine

2008

20. 8. – 25. 8. **2nd ConTEXt user meeting**
Bohinj, Slovenia
http://meeting.contextgarden.net/2008/

13. 9. **Herbsttagung**
und 39. Mitgliederversammlung von DANTE e.V.
Universität Tübingen
http://www.dante.de/dante/events/mv39/

18. 10. **GuiT** 2008 meeting
Pisa, Italien
http://www.guit.sssup.it/GuITmeeting/2008/2008.en.php/

(Fotos: Meike Schmedt und Karlheinz Geyer)

Stammtische

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

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

Berlin
Rolf Niepraschk
Tel.: 0 30/3 48 13 16
`Rolf.Niepraschk@gmx.de`
Humboldt Universität Berlin,
Fachbereich Wirtschaftswissenschaften
Spandauer Straße 1
Zweiter Donnerstag im Monat, 19.00 Uhr

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

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

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

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

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

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

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

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

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

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

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

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

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

Trier
Martin Sievers
stammtisch-trier@texberatung.de
Fetzenkneipe (Haus Fetzenreich)
Sichelstraße 36 (beim Sieh-Um-Dich)
54290 Trier
Dritter Montag des Monats, 20.15 Uhr

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

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

Würzburg
Bastian Hepp
dante@sning.de
nach Vereinbarung

Adressen

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

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

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

Präsidium

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

Server

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

Autoren/Organisatoren

Wolfgang Engelmann [4]
engelmann@uni-tuebingen.de

Jürgen Fenn [49]
Friedensallee 174/20
63263 Neu-Isenburg
juergen.fenn@gmx.de

Ulrike Fischer [7]
Bismarckstr. 91
41061 Mönchengladbach
skak@nililand.de

Klaus Höppner [4]
siehe Seite 58

Markus Kohm [48]
Freiherr-von-Drais-Straße 66
68535 Edingen-Neckarhausen
markus.kohm@gmx.de

Volker RW Schaa [4]
siehe Seite 58

Herbert Voß [3]
Wasgenstraße 21
14129 Berlin
Herbert.Voss@FU-Berlin.de

Uwe Ziegenhagen [36]
Humboldt-Universität zu Berlin
Wirtschaftswissenschaftliche Fakultät
ziegenhagen@wiwi.hu-berlin.de

Die TEXnische Komödie

20. Jahrgang Heft 3/2008 August 2008