Why Didn't METAFONT Catch On?

Dave Crossland
University of Reading, UK
dave@lab6.com
http://www.understandinglimited.com

Abstract

"There are three kinds of people: Those that can count, and those that can't."

METAFONT is an algebraic programming language for describing the shapes of letters, designed and implemented by Knuth as part of the original TEX typesetting system. It was one of the earliest digital type design systems, and is completely capable of dealing with the letters of any writing system, has always been freely available, and is remarkably powerful. Yet it never caught on with type designers.

The idea of software freedom has not caught on with many type designers either. Although the free software movement has produced a massive amount of high quality programs, there are few fonts and fewer original type designs — although there are some, with many available on CTAN.

Psychologists since Jung have suggested that human personality can be classified into 'types.' In "Please Understand Me II" David Keirsey relates 20th century theories of personality to the four humours proposed by Hippocrates and personified by the Greek gods Dionysus, Apollo, Prometheus, and Epimethius.

Do type designers tend to be of a certain humour? Is the inherently abstract METAFONT approach of describing shapes with algebra, or the rational appeal of software freedom, unintuitive for that temperament? Are people who do find free software and algebra intuitive involved in type design?

This presentation introduces the discipline of type design, reviews some of the fonts currently distributed in CTAN and explains their various origins. It draws attention to how they have been developed: The stroke – based METAFONT approach, involving writing source code, is starkly different to the typical outline – based 'Fontographer' approach, where letter shapes are visually and interactively designed. It suggests the technical benefits and drawbacks of each approach, and why some people prefer one approach over another.

Finally, it covers some tips and tricks for those contributing more free fonts to the TEX world, including some of the legal issues, where and how to source revivals, how to create new designs, and finally, how METAFONT might be modernised in a hybrid stroke/outline system to suit every personality.

Dave Crossland is an international public speaker on software freedom and fonts, runs a small business doing type design, information design and free software consultancy projects, and is a member of the UKTUG Committee. He is currently undertaking the MA Typeface Design programme at the University of Reading's Department of Typography.

Why we need LaTeX3

Jonathan Fine
LTS Strategic, The Open University
J.Fine@open.ac.uk
http://jonathanfine.wordpress.com

Abstract

The LaTeX3 project started in 1992. Since then, much has changed. XML has replaced SGML, and along with X/HTML has become the dominant markup language. CSS has replaced explicit style attributes in HTML pages, and is now a widely understood and used language for specifying design. Internet access is considerably more widespread, the web has gone from 1.0 to 2.0, Microsoft has replaced IBM, Linux went from nothing in 1991 to an open-source standard, and Google is on track to replace Microsoft.

In 1997 the LaTeX3 project said that LaTeX3 would provide:

- A new *input document syntax*, that aligns with SGML/XML.
- A new *class file interface*, that aligns with SGML/XML.
- A new *style-designer interface* that can work with a *visually-oriented, menu-driven specification system.*
- An *effective interactive help system* for document authors.
- *Thoroughly documented* and *modular* source code.

These goals are still worth achieving. This talk will focus on some recent progress, and in particular:

- Use of key-value syntax within tags.
- Separation of parsing from processing.
- An improved development environment.
- On-line interactive help systems.

MathTran and TEX as a web service

Jonathan Fine
LTS Strategic, The Open University
J.Fine@open.ac.uk
http://jonathanfine.wordpress.com

Abstract

In 2006/7 I developed and set up the public MathTran web service `www.mathtran.org`. This was done with funding from JISC and the Open University. It provides translation of TEX-notation formulas into high-quality bitmaps. In April 2008 it served 48,000 images a day for a growing range of sites. After tuning, the server could provide about 2 million images a day. It takes about 10 milliseconds to process a small formula.

MathTran images contain rich metadata, including the TEX source for the image and the `dvi` and `log` outputs due to that source. This makes it straightforward to edit and resize such images, or convert them to another format, such as SVG or PostScript.

MathTran, used with JavaScript, makes it considerably easier to put mathematics on a web page. In particular, the author of the page does not need to install any special software, and does not have to store thousands of image files.

The MathTran project is now focussed on the authoring of mathematical content. It has produced a prototype instant preview document editor. Funded by the 2008 Google Summer of Code, Christoph Hafemeister is developing JavaScript to provide autocompletion for commands and analysis of TEX errors, all integrated with an online help system embedded in the web page. Separate work is focussed on developing MathTran plugins for WYSIWYG editor web-page components.

This talk will present progress and prospects. It will also discuss some of the broader implications for the TEX community and software, such as

- People using TEX without installing TEX on their machine.
- Help components for web pages.
- Integration with third-party products.
- Standards for TEX-notation mathematics.
- Learning and teaching TEX.

LuaTEX: the TEX-Lua mix

Hans Hagen

Abstract

Now that the team is up speed and has released the first version of LuaTEX, it's time to reflect upon the way Lua will affect users and macro writers. This talks presents some examples of mixed usage, discusses the principles behind the interaction and reflects the conceptual differences between these two languages.

MPlib: an example of integration

Hans Hagen

Abstract

This talk discusses the way MPlib is integrated in the next generation of ConTEXt, tagged MkIV. The presentation will focus on the different aspects of integration as well as performance. The impact on workflows where heavy use is made of MetaPost integration will be discussed.

Image handling in LuaTEX

Hartmut Henkel
von Hoerner & Sulger GmbH, Schwetzingen, Germany
hartmut_henkel@gmx.de

Abstract

The Lua language allows to define new variable types, and LuaTEX uses this concept for new types like 'node' and 'font'. In this talk an image library as part of the LuaTEX engine is presented, which is built around a new 'image' type, giving extended image handling and embedding capabilities. The image primitives inherited from pdfTEX are still fully functional for compatibility.

First the process of image embedding and its limitations using the pdfTEX primitives is described. Then, after a short introduction about Lua libraries, the 'image' type of LuaTEX is presented together with the set of new Lua functions for image handling, and their use is illustrated by examples. As work is still ongoing, possible future extensions are discussed as well.

The galley Module or: How I Learned to Stop Worrying and Love the Whatsit

Morten Høgholm
LaTeX Project
morten.hoegholm@latex-project.org
http://www.latex-project.org

Abstract

TeX has a well-deserved good reputation for its line breaking algorithm and it has found its way into other software over the years. When it comes to inter-paragraph material such as penalties, skips and whatsits, things start getting murky as TeX provides little help in this area, especially on the main vertical list where most of the action is.

This article describes the galley module which seeks to control line breaking as well as taking care of inter-paragraph material being added at the right time. In other words, galley can assist packages such as breqn which has to construct paragraph shapes on the fly while taking current ones into account as well as ensuring the output routine doesn't get tricked by penalties, skips and whatsits appearing in places where they could allow breakpoints where none are intended.

LuaTEX: what has been done, and what will be done

Taco Hoekwater

Abstract

At TUG 2007 in San Diego, the first beta version of LuaTEX was presented. This year the team presents a version where significant parts of the TEX-Lua api are stable. This talk will give an overview of the components that make up LuaTEX: what libraries do we have and what callbacks are available. The team has some ideas about the next stages of development and these will be presented as well.

MPlib: the project, the library and the future

Taco Hoekwater

Abstract

The first stage of the MPlib project has resulted in a library that can be used in for instance LuaTEX, and is also the core of the MetaPost program itself. This talk will present the current state of affairs, the conversion process (from pascal to c), and the interface. A roadmap towards MegaPost will be presented too.

Languages for Bibliography Styles

Jean-Michel HUFFLEN
LIFC (EA CNRS 4157)
University of Franche-Comté
16, route de Gray
25030 BESANÇON CEDEX
FRANCE
hufflen@lifc.univ-fcomte.fr
http://lifc.univ-fcomte.fr/~hufflen

Abstract

BibTeX is the most used bibliography processor in conjunction with LaTeX. To put bibliography styles in action, it uses a stack-based language written with postfixed notations. Recently, other approaches have been proposed: some use a programming language for designing bibliography styles, e.g., Perl in Bibulus; some are based on converters to XML texts and use XSLT for bibliography styles; a recent proposal — the `biblatex` package — consists of using LaTeX command to control the layout of generated references... We propose a comparative study of these approaches and show which programming styles are encouraged, from a point of view related to methodology. Finally, we explain how this study has influenced the design of MlBibTeX, our multi-lingual reimplementation of BibTeX.

Data mining: Role of TEX files

Manjusha S. Joshi
Bhaskaracharya Institute in Mathematics, Pune, India.
manjusha.joshi@gmail.com

Abstract

Now a days most of the journals accepts TEX files from authors. Journals submits pdf files of the article or abstract of the article on the web site.

User wants to search for typical word, concept in the repository. Is it possible to faster the search because of availability of the TEX files? Since TEX files are structured documents. We can extract most of the important words from the document and make it available to user in forms of groups of related words. This will make user to search faster and get the required document in minimum time.

Smart ways of drawing PSTricks figures

Manjusha Joshi
Bhaskaracharya Institute in Mathematics
manjusha.joshi@gmail.com
~

Abstract

There are software with the help of which we can draw diagrams easily. However, there may be some things missing in it, like angle marks, arrows in between vertices etc. Being LaTeX user we also wish to use LaTeX math mode to label diagrams, for that one may require to edit the diagram.

Now-a-days, many figure drawing tools has option of exporting file in PSTricks. Which then generates, PSTricks code for the figure, in a simple LaTeX format. Which is readable and editable. It makes the job easy and efficient.

TEXworks: lowering the barrier to entry

Jonathan Kew
SIL International
jonathan_kew@sil.org

Abstract

One of the most successful TEX interfaces in recent years has been Dick Koch's award-winning TeXShop on Mac OS X. I believe a large part of its success has been due to its relative simplicity, which has invited new users to begin working with the system without baffling them with options or cluttering their screen with controls and buttons they don't understand. Experienced users may prefer environments such as iTeXMac, AUCTeX (or on other platforms, WinEDT, Kile, TeXmaker, or many others), with more advanced editing features and project management, but the simplicity of the TeXShop model has much to recommend it for the new or occasional user.

Besides the relatively "clean" interface, the second major factor in TeXShop's success is probably its PDF-centric workflow, with pdfTeX as the default typesetting engine. PDF is today's *de facto* standard for fully-formatted pages, and every user knows what a PDF file is and what they can do with it. Bypassing DVI reduces the apparent complexity of the overall process, and so reduces the "intimidation factor" for a newcomer. But TeXShop is built on Mac OS X-specific technologies, and is available only to Mac users. There does not seem to be an equivalent tool available on other platforms; there are many TEX editors and environments, but none with this particular focus.

This is the background to TEXworks, which aims to provide a simple TEX environment based on modern standards — including Unicode encoding and PDF output by default — with an uncluttered interface that does not overwhelm the newcomer. It is built using cross-platform, open-source tools and libraries, so as to be available on all today's major operating systems, with a native "look and feel" for each.

TEXworks also works with the new SyncTEX extension developed by Jérôme Laurens for pdfTeX and XeTeX, and shipping in TEX Live 2008. This provides a two-way link between source text and PDF output without requiring any modifications to the source document or support at the macro package level, bringing a new level of convenience to source/preview navigation.

This presentation will discuss the goals and features of the TEXworks project, describe the implementation that has been developed, and show the capabilities of the initial release.

The TEX community is invited to participate in the ongoing development of this environment, either at the actual code level or in areas such as document templates or interface localization.

Minion Math – The Design of a New Math Font Family

Johannes Küster
typoma GmbH
jk@typoma.com
http://www.typoma.com

Abstract

"Minion Math" is a set of mathematical fonts I have developed over the past 6 years. Designed as an add-on package to the Adobe MinionPro fonts, it consists of 20 OpenType fonts (4 weights, times 5 optical sizes). In future releases it will cover the complete Unicode math symbols, and more.

In the design I tried to avoid all flaws and shortcomings of other math fonts, with the aim of creating the most comprehensive and versatile set of math fonts to date.

In this presentation, I will talk about the design principles for Minion Math, and the design decisions I took. I will also show many samples of the fonts and will compare them to other math fonts as well.

Direct and reverse synchronization with SyncTEX

Jérô me LAURENS
Université de Bourgogne
jerome.laurens@u-bourgogne.fr

Abstract

In this presentation, we will focus on SyncTEX, a new synchronization technology now embedded into all the major TEX engines available in TEX-Live. This new design is extremely efficient and strong, such that old packages like `pdfsync` and `srcltx`, are now completely outdated. But this is not the only advantage: SyncTEX also brings pdf synchronization to the whole Unix world.

Multiple Simultaneous Galleys: A Simpler Model for Electronic Documents

Blanca Mancilla
School of Computer Science and Engineering
The University of New South Wales
UNSW SYDNEY NSW 2052, Australia
mancilla@cse.unsw.edu.au

John Plaice
School of Computer Science and Engineering
The University of New South Wales
UNSW SYDNEY NSW 2052, Australia
plaice@cse.unsw.edu.au

Toby Rahilly
School of Computer Science and Engineering
The University of New South Wales
UNSW SYDNEY NSW 2052, Australia
tobyr@cse.unsw.edu.au

Abstract

We present a general model for electronic documents supporting parallel containers of content, tied together through link components. This model is usable for a wide range of documents, including simple textual documents with footnotes and floats, complex critical editions with multiple levels of footnotes and critical apparatus, maps with multiple layers of visual presentation, and music scores.

This model is inspired from the C++ Standard Template Library, whose basis is that Containers + Iterators + Algorithms = Programs. In our approach, the "iterators" are pointers into the parallel containers, keeping track of callouts for notes, floats, and parallel links.

The data structures required for this model are remarkably simple, and will allow the rapid development of many different kinds of algorithms.

A Newbie's Experiences with Lilypond, Lilypond-book, LaTeX, and Perl

Joe McCool
Southern Regional College
mccoolj@src.ac.uk
http://benburb.demon.co.uk/apache2-default/joe.html

Abstract

The author is an active Irish traditional musician. He is also a keen inland boater. He is having a lot of fun composing a book on "Traditional Music for Boaters".

In this paper he describes his successes and frustrations using Lilypond, Lilypond-book, LaTeX, and ABC musical notation. Lilypond and LaTeX have a lot in common. Neither are WYSIWYG, neither demand GUIs. Both compile simple flat files to produce beautiful graphical output.

Lilypond's original manifestations produced output directly for LaTeX, but of late users writing books have been encouraged to use Lilypond-book. This looks for Lilypond code within LaTeX source files and produces graphics and associated instructions which can then be processed by LaTeX.

Most joy has been gained from automating these processes under Linux and Perl.

Mojca Miklavec
Faculty of Mathematics and Physics, University of Ljubljana,
mojca.miklavec@gmail.com
Arthur Reutenauer
GUTenberg, France
arthur.reutenauer@normalesup.org

Putting the cork back on the bottle: Improving Unicode support in TEX extensions

In the TEX world, the name of Cork is associated with a standardization effort dating back to 1990, the Cork font encoding, which can be used for most European languages written in the Latin script. At about the same time, though, a much wider standardization effort was initiated, as the Unicode Consortium was created to devise a universal character set suitable for any language and writing system. Of course, it wasn't long before people felt the need to support Unicode in TEX-like systems.

How far are we today? The latest extensions to the TEX engine are all labelled as "supporting Unicode", but upon closer inspection this reveals rather imprecise: does it mean enabling UTF-8 input, handling multibyte characters, or implementing all the Unicode character properties and algorithms?

In the framework of Google Summer of Code, one of us (Arthur) is sponsored to improve Unicode support in TEX. The original accepted proposal was about three aspects: combining characters, bidirectional algorithm and line-breaking properties, and I will be working on LuaTEX and XeTeX, who both handle multibyte characters and UTF-8 characters natively.

Combining characters are Unicode's diacritical marks: you put them after a character (called "base character") to add an accent to it. This is very similar to TEX's `\accent` primitive, except that they come *after* the character they apply to, and that you can stack them. There are also equivalences between sequences containing combining characters and *precomposed* characters, and algorithms to transform them (called normalization).

The bidirectional algorithm ("bidi" for short) specifies how to handle text mixing characters from writing systems with different directions, for example, English and Arabic.

Finally, Unicode characters have properties pertaining to line breaks: for example, characters like NBSP forbid breaks after them (very much like TEX's "tie"); other allow break under certain conditions, etc.

Those three aspects were chosen because it was felt they all related to capabilities that TEX had from the beginning and that they were, therefore, among the most interesting aspects of Unicode with respect to TEX processing.

In the mean time, Mojca has initiated an effort to convert TEX Live's hyphenation patterns to UTF-8. The challenge here is to integrate the current pattern files in a way that they can be used with both 8-bit engines (TEX, pdfTEX), and engines supporting native UTF-8 input (LuaTEX, XeTeX). It has uncovered interesting issues of the relationship between TEX and Unicode: it helps us understand to which extent, and how, TEX input can be converted to Unicode, and vice-versa.

Windows of opportunity: A (biased) personal history of two decades of LaTeX development—Are there lessons to be learned?

Frank Mittelbach
LaTeX Project
frank.mittelbach@latex-project.org
http://www.latex-project.org

Abstract

Looking back at twenty-odd years involvement in LaTeX development and maintenance the author highlights the (in his opinion) most important milestones and pitfalls.

- What are significant events that came at the right moment?
- Which important events came at the wrong moment?
- What were the biggest failures and why?

From this data the article attempts to draw conclusions as to how the future of LaTeX could be shaped in a way beneficial to everybody involved and what needs to happen to make this possible.

Advanced features for publishing mathematics, in PDF and on the Web

Ross Moore
Macquarie University, Sydney, Australia
ross@maths.mq.edu.au
http://www.maths.mq.edu.au/staff/ross.html

Abstract

Increasingly mathematical, scientific and technical information is being distributed by electronic means, but having a high-quality paper printout remains important. I will show examples of techniques that are available for having both high-quality typesetting, in particular of mathematics, as well as useful navigation features and text-extraction within electronic documents.

For HTML, some examples will be shown of the use of jsMath within webpages for mathematics journals and for conference abstracts. With PDF, as well as the usual bookmarks and internal hyperlinks for cross-references and citations, advanced features include: (i) Metadata attachments; (ii) copy/paste and searching for mathematical symbols or the underlying TEX coding; (iii) popup images of (floating) figures and tables; (iv) mathematical symbols within bookmarks; (v) bookmarks for cross-referenced locations.

A further feature, particularly useful with mathematics papers, is the ability to make batched searches of the American Math. Society's *MathSciNet* database, allowing hyperlinks to be generated for most bibliography entries.

A Pragmatic Toolchain: TEX and Friends and Friends of Friends

Steve Peter
Pragmatic Programmers
steve@pragprog.com
http://www.pragprog.com

Abstract

In this talk, we present the toolchain used to produce the award-winning Pragmatic Bookshelf titles (http://www.pragprog.com) and examine some of the pleasures and pitfalls encountered using TEX, XML, XSLT, Ruby and other open technologies.

Creating cuneiform fonts with METATYPE1 and FontForge

Karel Píška
Institute of Physics, Academy of Sciences
Prague, Czech Republic
piska (at) fzu dot cz

Abstract

The cuneiform font collection covers the Basic Latin (ASCII) block and glyph subsets for Akkadian, Ugaritic and Old Persian with current total number about 600 cuneiform signs. An extension for other languages is planned (Neo-Babylonian, Hittite, etc.). All cuneiform sign forms visually correspond to uniform "Neo-Assyrian" shapes.

Fonts are produced in two steps. With METATYPE1, the package developed by the authors of the Latin Modern and TEXGyre fonts, we can generate hundreds (or thousands) glyphs to assembly a Type 1 font with many glyphs, but with no predefined encoding. The older variant of the cuneiform font collection, made 10 years ago, consists of several separate Type 1 components. A relative small number of mostly simple and repetitive elements is described by METAPOST macros in three variants.

In the second step we construct OpenType using FontForge, the free font editor, created by George Williams. Cubic and quadratic approximation of outline curves are allowed because of a simple design of cuneiform wedges. Therefore both TTF flavored and PostScript (CFF) flavored formats may be generated. We use FontForge scripting facilities, it is also possible to write commands in its internal textual format (SFD), directly or with some pseudo-automatic tools.

Unfortunately, the glyph repertoire does not correspond to Unicode because more than 300 glyphs do not have their Unicode numbers, and, on the other hand, my fonts covers only about 20% of the Unicode Sumerian-Akkadian cuneiform range (cuneiform signs and numeric signs). The paper describes cuneiform design and a process of font development.

Multidimensional Text

John Plaice
School of Computer Science and Engineering
The University of New South Wales
UNSW SYDNEY NSW 2052, Australia
plaice@cse.unsw.edu.au

Blanca Mancilla
School of Computer Science and Engineering
The University of New South Wales
UNSW SYDNEY NSW 2052, Australia
mancilla@cse.unsw.edu.au

Toby Rahilly
School of Computer Science and Engineering
The University of New South Wales
UNSW SYDNEY NSW 2052, Australia
tobyr@cse.unsw.edu.au

Abstract

The Unicode model of text makes a clear distinction between character and glyph, and in so doing, paradoxically, creates the impression that the ultimate representation for text is some form of abstraction from its visual presentation. However, the level of abstraction for different languages encoded "naturally" in Unicode is quite different.

We propose instead that text be encoded as sequences of context-tagged indices into arbitrary indexed structures, including not just character sets such as Unicode, but also dictionaries of words or compound words. Furthermore, these sequences need not necessarily contain elements from the same indexed structures.

Using our approach allows natural solutions for a wide range of problems, including the creation of documents that can be printed using several alternate spellings, the automatic generation of error messages with arguments, and the correct generation of nouns or adjectives with number, case or gender markers or of verb conjugations.

docx2tex

Krisztián Pócza
Eötvös Loránd University, Faculty of Informatics, Department of Programming Languages and Compilers, Pázmány Péter sétány 1/C. H-1117, Budapest, Hungary
kpocza@kpocza.net
http://kpocza.net/

Mihály Biczó
Eötvös Loránd University, Faculty of Informatics, Department of Programming Languages and Compilers, Pázmány Péter sétány 1/C. H-1117, Budapest, Hungary
mihaly.biczo@t-online.hu
http://avalon.inf.elte.hu/personal/hdbiczo/

Zoltán Porkoláb
Eötvös Loránd University, Faculty of Informatics, Department of Programming Languages and Compilers, Pázmány Péter sétány 1/C. H-1117, Budapest, Hungary
gsd@elte.hu
http://gsd.web.elte.hu/

Abstract

Docx2tex is a small command line tool that uses standard technologies to help users of Word 2007 to publish publications where typography is relevant or only papers produced by TEX are accepted. Behind the scenes, docx2tex uses common technologies to interpret Word 2007 OOXML format without utilizing the API of Word 2007. Docx2tex is planned to be published as a free open source utility that is accessible and extensible by everyone. This paper has been originally written in Word 2007 and then converted to TEX using docx2tex.

Parallel Typesetting

Toby Rahilly
School of Computer Science and Engineering
The University of New South Wales
UNSW SYDNEY NSW 2052, Australia
tobyr@cse.unsw.edu.au

John Plaice
School of Computer Science and Engineering
The University of New South Wales
UNSW SYDNEY NSW 2052, Australia
plaice@cse.unsw.edu.au

Blanca Mancilla
School of Computer Science and Engineering
The University of New South Wales
UNSW SYDNEY NSW 2052, Australia
mancilla@cse.unsw.edu.au

Abstract

We present the general mechanism by which logical content, arranged in multiple interacting containers, can be typeset into a set of visual substrates. The overall algorithm is iterative, with the successive iterations refining a multidimensional context that parameterises the behavior of the algorithm.

Each iteration consists of three parts. First, each visual substrate is informed which parts of which logical containers are to be placed thereon. Second, in parallel, the content placed in the substrates is typeset. Third, the resulting layout in each substrate is assessed for goodness, thereby resulting in the refinement to the overall context.

In the talk, we will present the theory and the practice behind this algorithm.

Three typefaces for mathematics

Daniel Rhatigan
The University of Reading
sparky@ultrasparky.org
http://www.ultrasparky.org/work

Abstract

This paper examines the issues involved in the design of typefaces for mathematics. After a brief discussion of some of the typographic and technical requirements of maths composition, three case studies in the development of maths types are presented: Times 4-line Mathematics Series 569, a complement to the Times New Roman text types as set with Monotype equipment; AMS Euler, an experimental design intended to contrast against non-mathematical typefaces set with TEX; and Cambria Math, designed in concert with a new text face to take advantage of new Microsoft solutions for screen display and maths composition.

In all three cases, the typefaces were created to show the capabilities of new technological solutions for setting maths. The technical advances inherent in each font are shown to be as central to its function as its visual characteristics.

By looking at each typeface and technology in turn, and then comparing and contrasting the issues that are addressed in ! each case, it becomes apparent that even though certain challenges are overcome with technical advances, the need to consider the specific behaviours oftype in a maths setting remains constant.

Writing Gregg Shorthand with LaTeX and METAFONT

Stanislav Šarman
Computing Centre
Clausthal University of Technology
Erzstr. 51
38678 Clausthal
Germany
Sarman@rz.tu-clausthal.de
http://www3.rz.tu-clausthal.de/~rzsjs/steno/Gregg.php

Abstract

We present an on-line system, which tries to convert English text into Gregg shorthand(O'Kennedy, 1990), phonetic pen writing system used in the U.S. and Ireland.

The given text is at first tokenized (tokens being punctuation marks, words and common phrases). For each of the tokens a METAFONT glyph is generated on-the-fly. These glyphs are (smooth) combinations of shorthand graphems corresponding to phonems obtained from a pronunciation lexicon (Fitt, 2006). The shorthand text is then set with LaTeX.

References

Fitt, Susan. "Unisyn multi-accent lexicon". 2006. http://www.cstr.ed.ac.uk/projects/unisyn/.

O'Kennedy, Gerard. *Gregg Shorthand Manual Simplified*. McGraw-Hill, 1990.

xindy Revisited – Multi-Lingual Index Creation for the UTF-8 Age

Joachim Schrod
Net & Publication Consultance GmbH
Kranichweg 1
63322 Rödermark
Germany
jschrod@acm.org
http://www.xindy.org/

Abstract

xindy is an index processor. Just like MakeIndex, it transforms raw index information into a sorted index, made available as document text with markup that may be processed by TEX to produce typeset book indexes. Unlike MakeIndex, it is multi-lingual and supports UTF-8 encoding, both in the raw index input and in the tagged document output.

xindy draws its strengths from five key features.

1. *Internationalization* is the most important feature point and was originally xindy's raison d'être: with the standard distribution, xindy knows how to handle 53 languages and dialects correctly out of the box.
2. *Markup normalization* and *encoding support* is the ability to handle markup in the index keys in a transparent and consistent way, as well as different encodings. Predefined encodings are not only UTF-8 to support XETEX, also supported is LICR, the encoding that's output by standard LATEX to its raw index files, and TEX/Omega's low-level output of (Unicode) characters.
3. *Modular configuration* enables the reusability of index configurations. For standard indexing tasks, LATEX users do not have to do much except to use available modules.
4. *Location references* go beyond page numbers. An index entry points to a location in the main text. While most index processors can work only with numbers, xindy features a generalized notion of location references that can be book names, law paragraphs, URLs and other references.
5. *Highly configurable markup* is another cornerstone. While this is usually not as important for LATEX users, it comes in handy if one works with other author systems besides TEX.

While development of xindy has been dormant for quite some time, the last few months saw a flurry of renewed energy and new work to get xindy in the hand of its potential users. The distribution has been streamlined and is now available in standardized source form, thus paving the road for a future acceptance into TEX-Live.

In addition, TUG2008 at Cork is the perfect avenue to present xindy's *TUG30 release*, that will feature for the first time full support for Linux/Unix, Mac OS X, and Microsoft Windows and also straight-forward and simple installation possibilities, both as binary and source distribution.

DocScape Publisher
Doing layouts in XML

Martin Schröder

Abstract

The DocScape Publisher from QuinScape GmbH has its focus on data based publishing of input in XML form.

In its core, currently LaTeX, David Carlisle's XMLTeX, and pdfTeX are employed extensively.

We show how rule-based layouts are done and compare this to the LaTeX way.

PDF libraries and TEX

Martin Schröder

Abstract

One of the reasons for the success of pdfTEX was the quality of the PDF inclusion, which uses code from XPDF.

Over the last years some (free) PDF libraries and tools have been developed. I will show some of these and discuss their usability with(in) TEX.

Five years of pdfTEX– lessons learned

Martin Schröder

Abstract

From 2003 till 2008 I was the maintainer of pdfTEX.

I will share some ideas about open source projects learned during this time.

Meta-designing parameterized Arabic fonts for AlQalam

Ameer M. Sherif, Hossam A. H. Fahmy
Electronics and Communications Department
Faculty of Engineering, Cairo University, Egypt
ameer dot sherif (at) gmail dot com, hfahmy (at) arith dot stanford dot edu
http://arith.stanford.edu/~hfahmy

Abstract

In this paper we discuss in detail how parameterized Arabic letters are meta-designed using METAFONT and then used in forming words. Parameterized Arabic fonts enable greater flexibility in joining glyphs together, rendering words with imperceptible junctions and smoother letter extensions. This work aims at producing written Arabic with quality close to that of calligraphers. Words produced using our parameterized font are compared to other widely used fonts in a subjective test and results are presented.

Metadata extraction from LaTeX documents

Pter Szab
Budapest University of Technology and Economics
pts@fazekas.hu
http://www.inf.bme.hu/~pts/meta_from_latex/

Abstract

Metadata such as author, title and number of pages in a LaTeX document can be useful outside the document, such as in the table of contents of a journal, in a bibliography or in a web index of papers. TeX provides the `\write` primitive to write tokens to files, but further cleaning and parsing is necessary to make this information useful outside LaTeX.

We address these issues and present a framework which combines LaTeX and Ruby code to extract metadata from the articles in the *Periodica Polytechnica* journal, and convert it to XML with Unicode text, for submission to web indices. The framework also provides automatic formatting such as typesetting the author list of a paper with some of the authors sharing a common affiliation.

Observations of a TEXnician for Hire

Boris Veytsman
Computational Materials Science Center, MS 6A2
George Mason University
Fairfax, VA 22030
borisv (at) lk dot net

Abstract

Several years ago the author was tempted by extremely cheap rates for *TUGboat* advertisements, and declared *urbi et orbi* he was a TEX consultant. This audacious step lead to many interesting experiences. Some results of this work were published on CTAN and listed at `http://borisv.lk.net/latex.html` (the list includes both commissioned packages and the ones I wrote for my own purposes).

In this talk I tell about my past projects, big and small, and discuss the lessons learned from my journeys in the fascinating world of publishers, editors and authors. I describe writing book and journal styles, communication with customers and other issues relevant for TEX consulting.

Medical Pedigrees with TEX and PSTricks: New Advances and Challenges

Boris Veytsman
Computational Materials Science Center, MS 6A2
George Mason University
Fairfax, VA 22030
borisv (at) lk dot net

Leila Akhmadeeva
Bashkir State Medical University
3 Lenina Str. Ufa, 450000, Russia
leila_ufa (at) mail dot ru

Abstract

A medical pedigree is an important tool for researchers, clinicians, students and patients. It helps to diagnose many hereditary diseases, estimate risks for family members, etc (Bennett, Steinhaus, Uhrich, O'Sullivan, Resta, Lochner-Doyle, Markei, Vincent, and Hamanishi, 1995). Recently we reported a comprehensive package for automatic pedigree drawing (Veytsman and Akhmadeeva, 2006; Veytsman and Akhmadeeva, 2007). Since then we extended the algorithm for a number of complex cases, including correct drawing of consanguinic relationships, twins and many others.

In this talk we review the facilities of the current version of the program and the new challenges in computer-aided drawing of medical pedigrees.

We try to make the talk interesting to TEXnicians and TEXperts by discussing the experience of design a TEX-based application working in a "real world".

References

Bennett, Robin L., K. A. Steinhaus, S. B. Uhrich, C. K. O'Sullivan, R. G. Resta, D. Lochner-Doyle, D. S. Markei, V. Vincent, and J. Hamanishi. "Recommendations for Standardized Human Pedigree Nomenclature". *Am. J. Hum. Genet.* **56**(3), 745–752, 1995.

Veytsman, Boris, and L. Akhmadeeva. "Drawing Medical Pedigree Trees with TEX and PSTricks". *PracTEX J.* (4), 2006. https://www.tug.org/pracjourn/2006-4/veytsman.

Veytsman, Boris, and L. Akhmadeeva. "Drawing Medical Pedigree Trees with TEX and PSTricks". *TUGboat* **28**(1), 100–109, 2007. http://www.tug.org/TUGboat/Articles/tb28-1/tb88veytsman-pedigree.pdf.

Do we need a Cork math font encoding?

Ulrik Vieth
Vaihinger Straße 69
70567 Stuttgart
Germany
ulrik dot vieth (at) arcor dot de

Abstract

The city of Cork has become well-known in the TEX community, ever since it gave name to an encoding developed at the European TEX conference of 1990. The Cork encoding, as it became known, was the first example of an 8-bit text font encoding that appeared after the release of TEX 3.0, which was later followed by a number of other encodings based on similar design principles.

As of today, the Cork encoding represents one of several possible choices of 8-bit subsets from a much larger repertoire of glyphs provided in font projects such as Latin Modern or TEX Gyre. Moreover, recent developments of new TEX engines are making it possible to take advantage of OpenType font technology directly, largely eliminating the need for 8-bit font encodings altogether.

During all the time since 1990, math fonts have always been lagging behind the developments in text fonts. While the need for new math font encodings was recognized early on and while several encoding proposals have been discussed, none of them ever reached production quality and became widely used.

In this paper, we want to review the situation of math font encodings as of 2008, especially in view of recent developments of Unicode math fonts such as the STIX fonts or CambriaMath. In particular, we try to answer the question whether a Cork math font encoding is still needed to make use of such fonts in TEX or whether Unicode and OpenType font technology might eventually eliminate the need for TEX-specific math font encodings.