Bart Childs

LaTeX source from word processors

Hennings' CTAN survey is a good starting point when considering projects implied by the title of this article. I find it a fair view of most related packages. He suggests having one of two goals: converting the *document structure* or converting the *appearance*. My goal is neither of these: it is producing LaTeX source that is clean and therefore maintainable. This is in keeping with Knuth's original goals in producing TeX: graphic excellence and a document convenient for archiving. Structure and appearance are important but neither of these in the word processor format are as important as clean LaTeX to me. My current system is a hybrid. I use the common word processor exchange format — Rich Text Format, the OpenOffice Writer package and its `Writer2LaTeX` application, and macros for the Emacs editor written in Elisp. The test cases for this system are books on rotordynamics, a CS/1 programming text, a memoir on a friend's life including significant text fragments in the Czech language, and a novel that includes three love triangles. Even the worst case of significant mathematics formatting done in Word Perfect is tractable (I did not say easy). The most surprising problems are due to the limited skill of the users of their word processors.

Bruno Delprat & Stepan Orevkov

MayaPS: Typing Maya hieroglyphics with (La)TeX

We present a system for the hieroglyphical composition of ancient Maya texts, to be used for their palaeography, the production of dictionaries, epigraphical articles and textbooks. It is designed on the base of TeX and PostScript using the Dvips interface, and includes a set of Maya fonts.

The ancient Mayan writing system is very particular: base writing signs attach each other from all four sides (left, right, top, bottom), being also rotated and rescaled and could not be produced with usual TeX's tools. For example, we can type: `\maya{li.AM2 u.TUN/CHU uj.UJ.ki death.KIMI/la}` to obtain the example shown in the preprint.

Peter Flynn

A university thesis class: Automation and its pitfalls

Despite the large collection of thesis classes available, there are always features that an institution needs which can better be met by writing Yet Another Thesis Class. There are also variations in the quality and availability of documentation, assumptions about preloaded packages, and the ease (or otherwise) with which the author can modify the layout.

In the case of UCC, the official requirements were very simple, avoiding the tendency to over-specify details found in some university formats. The class was also required to be generally applicable to any discipline, so only a minimum of packages was needed (although this turned out to be more than anticipated).

The major component was an attempt to automate as much of the front matter as possible, based on options to tokenize the discipline and class of degree. This was done to avoid accidental omissions, variations, and misspellings in the titling (or even deliberate rewording); and to ensure that the relevant compulsory components appeared in the right place without the author having to do anything.

The result has been piloted with 20–30 PhD candidates for a year, and needs only a few final changes before release. Two other institutions in the state have already expressed an interest in basing their own thesis classes on this one.

Federico Garcia

Documentation in TeXnicolor

My package `colordoc` builds on Frank Mittelbach's `docstrip` system of documentation, adding some utilities to use color in the code: matching delimiters (`{` and `}`) are colored the same, just as matching `\if`–`\fi` pairs. Commands are made red, bold, and italics, when they are being `\def`ined, just as variables when they are being declared (`\newcount`, `\newif`, etc.). These tools have certainly saved me a lot of time and trouble when editing or trying to understand a code. In the presentation I also describe the interesting general lines of the workings of both `doc` and `colordoc`.

Federico Garcia

TeX and music

For some years I've been working on writing a professional music typesetting system with TeX (and METAFONT). In 2005 I even had a grant from the TeX Development Fund that allowed me to do a first model of the system. Since then the system has evolved, and particularly between 2010 and now I have actually developed a promising model.

This presentation does a little bit of history of the main idea of the TeXmuse system, which as will be seen is entirely inspired by the TeX 'spirit'. This touches on the potentially disastrous problems of WYSIWYM for music typesetting; on what similarly oriented systems have done (and not); and on whether even in spite of these problems the project is worth pursuing. The answer is in the affirmative, mainly because of the more disastrous problems of the alternative, i.e. commercial software. (In a nutshell, those problems are that those programs are *not* inspired by TeX's spirit! In concrete, there are some achievements of TeXmuse that would change the life of any composer.) I also demonstrate some pretty cool programming tricks

that I have found, both in TEX and in METAFONT, that in my view speak to the beauty of the systems. The talk does not require technical knowledge of music or music typesetting.

Jim Hefferon & Michael Doob
Reaching for the stars with Asymptote

Asymptote is a stand-alone program that excels at generating line art, and takes its inspiration from Metafont and MetaPost. It is relatively recent, but already quite capable. I'll introduce some of its features for a person who has never seen it at work.

Troy Henderson
User-friendly web utilities for generating LATEX output and MetaPost graphics

There are several facets of the creation of LATEX documents and MetaPost graphics that deter users from initially trying both LATEX and MetaPost. These include the basic structure of the source files, the compilation of the source files, and the conversion of the output to a desired format. Furthermore, many TEX users wish to create 2D and 3D graphs of functions for inclusion into their documents. Many of these graphs require considerable amounts of source code to create professional quality graphics, and this is yet another deterrent for those who might otherwise consider using MetaPost. This presentation will introduce several free web utilities that aim to eliminate each of these obstacles and describe the usage and methods of these utilities.

Amy Hendrickson
The wonders of `\csname`

A surprisingly useful tool, `\csname...\endcsname`, offers many opportunities for interesting and useful macros, whenever it is convenient to dynamically generate a series of definitions. When each definition contains a counter in its name, we can then call the definition using a loop that advances a counter, and then calling the definition using the loop counter inside the `\csname...\endcsname`.

A trivial use is for endnotes. More interestingly, csname definitions can be used to send a set of definitions to the `.aux` file, where each new definition contains the current page number in its name, with a 'security level' number being defined. This allows the dynamic redefinition of the security level for a particular page, *within the* `.aux` *file* depending on whether the new number is higher or lower than the previous number. This can be used to determine the highest security level on any particular page. When the `.aux` file is then input, we can access the csname definition in a running head of the LATEX document, calling `\csname...\endcsname` control sequence and using the current page number within `\csname...\endcsname`, to activate the definition of the highest security level on that page, and use the information to print the security level on top of the page.

Another interesting use is for on-line report generation, where a csname definition can be used, for instance, to generate hyperlinks for financial analyses of stocks, in a report that compares hundreds of stocks, and then be able to automatically build a hyperlinked table of contents, using tabs built with Ti*k*Z.

Code and examples will be shown for each of these methods and uses of dynamically generated macros using `\csname...\endcsname`, and we'll look at some other ways this tool may be used as well.

Richard Koch
The MacTEX install package

MacTEX installs everything needed to run TEX on a Macintosh with a single button click. I'll discuss the history of this package — Wendy's conspiratorial lunch and Jonathan Kew's all night coding session — modifications over the years, and important changes in the 2012 version. I'll discuss the importance of installing an unmodified, vanilla version of TEX Live, explain how we add extra pieces so everything is automatically configured and ready to go, and end with possible future changes. The talk may contain completely unmotivated and irrelevant additional comments.

David Latchman
Preparing your thesis in LATEX

The submission of a thesis or dissertation is the culmination of many a graduate student's career. Given the time and effort toward research and attaining their degrees, this can often be a stressful time for many students. LATEX offers the advantage of separating form from content and as the typical university thesis class can take care of a university's formatting requirements thus makes a student's life easier — well, at least it is supposed to.

Unfortunately, some formatting 'blends' into the content, thereby adding to the stress of an already unpleasant task. But there is some light at the end of the tunnel. With some preparation, typesetting a thesis in LATEX can be relatively pain free. But it's not simply a matter of just knowing what packages but *how* to use them and what is needed to use them effectively. Topics covered will include the typesetting of equations — both mathematical and chemical — as well as the proper formatting of tables and bibliographies.

Sherif Mansour & Hossam Fahmy
Experience with Arabic and LuaTEX

This is an experience report of an attempt to include the AlQalam font for Arabic script within

LuaTEX. We describe the problems we faced trying to figure out how to use a new right-to-left font within LuaTEX. We also describe how to call the many different shapes that are defined via parameters in the original font. We also present some ideas to modify the line breaking algorithm of TEX to allow the use of different shapes for the same character in order to justify the line. This is still work in progress.

Frank Mittelbach
E-TEX: Guidelines for future TEX extensions, revisited

Shortly after Don Knuth announced TEX 3.0 I gave a paper analyzing TEX's abilities as a typesetting engine. The abstract back then said:

> Now it is time, after ten years' experience, to step back and consider whether or not TEX 3.0 is an adequate answer to the typesetting requirements of the nineties.
>
> Output produced by TEX has higher standards than output generated automatically by most other typesetting systems. Therefore, in this paper we will focus on the quality standards set by typographers for hand-typeset documents and ask to what extent they are achieved by TEX. Limitations of TEX's algorithms are analyzed; and missing features as well as new concepts are outlined.

Now — two decades later — it is time to take another look and see what has been achieved since then, and perhaps more importantly, what can be achieved now with computer power having multiplied by a huge factor and last not least by the arrival of a number of successors to TEX which have lifted some of the limitations identified back then.

Bob Neveln & Bob Alps
Adapting ProofCheck to the author's needs

ProofCheck is a system for writing and checking mathematical proofs. Theorems and proofs are contained in a plain TEX or LATEX document. Parsing and proof checking are accomplished through Python programs which read the source file. Although the use of these programs has never been restricted to any particular logical or mathematical language, the work required to actually implement an author's choices in these matters, especially in the logic, and to make the necessary modifications of the supporting files has been sufficiently laborious as to pose an obstacle to the use of ProofCheck. This paper describes updates to the system whose purpose is to alleviate these labors to the extent possible so as to facilitate the use of ProofCheck in a logical and linguistic setting of the author's choice.

Steve Peter
Metafont as a design tool

Well-written Metafont sources provide a font designer with a nearly unparalleled tool to explore variations on a typographic theme. Paired with TEX in an advanced environment, the designer can explore serif structure, bracketing, weight variations and more in the context in which the font will be used: real textual matter. I'm going to ignore the production problems inherent to Metafont (not to mention the various possible solutions) to concentrate on the design aspects of this amazing tool.

Norbert Preining
Typesetting with Kanji — Japanese typography

Japanese typography is very particular and demanding in several respects: four different writing systems: Kanji, Hiragana, Katakana, Roman letters mixed together; vertical and horizontal typesetting; traditional grid layout versus mixture of writing systems. This all led to a spin-off TEX implementation called "Publishing TEX" (pTEX) that can deal with these specifics.

Until 2011 there was an independent distribution of TEX for Japanese users, first based on teTEX, later on TEX Live (`ptetex`, `ptexlive`). TEX Live 2011 and 2012 introduced all of the necessary tools and features and we hope that with TEX Live 2012 the need for a special setup for Japanese users is past.

In this talk we give an overview of the specialities of Japanese typography, presenting the difficulties met in modern texts. Continuing, we present solution provided by TEX Live to some of these problems, and discuss further development.

Norbert Preining
TEX Live 2012: Recent developments

TEX Live will be released in early summer 2012 and brings a couple changes that have been in the works for a long time: a "multi-`updmap`" that reads several `updmap.cfg` files, and multi-repository support for the TEX Live Manager tlmgr.

`updmap` is a program that generates the necessary configuration files for `dvips`, `dvipdfm(x)`, `pdftex`, and `pxdvi` to display PostScript Type 1 fonts. It reads a configuration file that lists several map files, and combines all the font definitions from these map files. Until now local font maps had to be integrated into this updmap.cfg file, and so could easily be overwritten or otherwise be lost.

The new implementation has a long history. The original Perl version was written by Fabrice Popineau for Windows, later extended by Reinhard Kotucha and Karl Berry and used starting last year on all platforms supported by TEX Live. The

code has now been extended to deal with multiple configuration files in a transparent way.

This allows a clear separation of `updmap.cfg` file parts. One `updmap.cfg` file now can (but does not have to) only provide information about the `texmf` tree it resides in. In other words, fonts installed into, for example, the `TEXMFLOCAL` tree can be activated by an entry in the `updmap.cfg` file *in this tree*.

We will discuss this new functionality and provide usage examples and advise on transition from the old system.

The other big change in TEX Live this year is the extension of the TEX Live Manager with the capacity of reading multiple repositories. In recent years, a few alternative TEX Live repositories have come into existence with a wide range of usage patterns: distribution of local packages (Japanese TEX related packages in `tlptexlive`, Korean TEX User Group repository), TEX Live infrastructure testing (in `tlcritical`), provision of development and nonfree packages (in `tlcontrib`), etc.

Until now a user had to go through all desired repositories one by one passing the necessary parameters for each in turn. The new `tlmgr` supports use of several sources at the same time. The selection of packages appearing in multiple repositories is done by "pinning" packages to a repository.

We will present this new functionality, give usage examples, and a guided tour through setting up and using this new feature.

We will close with an overview on other changes in TEX Live 2012.

Will Robertson & Frank Mittelbach
LATEX3: From local to global — a brief history and recent developments

The original source code for LATEX3 dates to the early 1990s. Key aspects of its development occurred during that decade, but it was not until the late 2000s that the project began delivering code that was widely used by mainstream LATEX users. What happened in this time? This talk will discuss how LATEX3 development evolved over the decades and how it reached a state of being used to produce real users' documents whether or not they are actually aware of it. LATEX3 can be thought to consist of separate 'layers', and the programming layer known as `expl3` is starting to be used to solve problems in and write packages for LATEX 2_ε. Our plans are not restricted to such 'under-the-hood' measures, however, and we have discussed layers of LATEX3 that will have more visibility at the user interface. Our talk will discuss these separate layers and where our plans lead in the future, and will conclude with a demonstration of what's new in the current code.

Will Robertson
The lineage and progeny of fontspec and unicode-math

My first LATEX package, `fontspec`, was written in 2004 before I knew how to program in LATEX and in truth before I knew how to program at all. This trial-by-fire introduced me to the lovely world of TEX programming and after some time I ended up writing a smattering of other works. (All the while actually starting to learn what this whole 'programming' thing was all about, including how to please and displease people who were just trying to get work done, thank you very much.) Some time later I foolishly tried 'planning' an ambitious new package, `unicode-math`, that took significantly longer to release. In the course of writing that package I learned really just how little I actually knew, and as a side-effect somehow ended up helping to write code for the LATEX3 project. In this talk I will talk about the motivation for writing these two packages, discuss recent developments with them, and finally touch on how LATEX3 influenced their development.

Herbert Schulz
Workshop: Introduction to TeXShop

A workshop introducing some of the more obscure and less used features of TeXShop for users who wish to become more proficient in its use to produce LATEX documents.

Christina Thiele
Almost 30 years of using TEX

It's not just TEX that's gotten older and more seasoned ... Reflections on changes in TeX and friends as used in a small typesetting company: software and hardware, of course, but also procedures and skills, resources that went from zero to virtually infinite, all of it interwoven with life and personal change. It's not earth-shaking news, but we've come far enough that looking back yields some interesting comparisons.

Didier Verna
Star TEX, the Next Generation

In 2010, I asked Donald Knuth why he chose to design and implement TEX as a macro-expansion system (as opposed to more traditional procedure calls). His answer was that: 1) he wanted something relatively simple for his secretary who was not a computer scientist; 2) the very limited computing resources at that time practically mandated the use of something much lighter than a true programming language.

The first part of the answer left me with a slight feeling of skepticism. It remains to be seen that TEX is simple to use, and when or where it is, its underlying implementation has hardly anything to do with it.

The second part of the answer, on the other hand, was both very convincing and arguably now obsolete as well. Time has passed and the situation today is very different from what it was 30 years ago. The available computing power has grown exponentially, and so has our overall skills in language design and implementation.

Several ideas on how to modernize TEX already exist. Some have been actually implemented. In this talk, I will present mine. Interestingly enough, it seems to me that modernizing TEX can start with grounding it in an old yet very modern programming language: Common Lisp. I will present the key features that make this language particularly well suited to the task, emphasizing on points such as extensibility, scriptability and multi-paradigm programming. The presentation will include reflections about the software engineering aspects (internals), as well as about the surface layer of TEX itself. Most notably, I will explore the possibilities of providing a more consistent syntax to the TEX API, while maintaining backward compatibility with the existing code base.

Boris Veytsman
TEX and friends on a Pad

TEX on an Eee Pad is quite workable.

Boris Veytsman & Leyla Akhmadeeva
Towards evidence-based typography: First results

At the previous TUG meeting we reported experimental design for checking whether the typographic features of the text (fonts, page layout, justification, etc.) influence the way readers comprehend and remember the contents. Our study is intended primarily for the designers of textbooks, where the comprehension of the text is very important.

In this work we report the first results of our study. It seems that despite the beliefs of typographers, the text comprehension and the speed of reading is not much influenced by typography. These findings confirm the generalized ecological hypothesis by Legge and Bigelow. It seems the human brains are flexible enough to allow us to read even badly designed pages.

We also discuss the role of TEX as a useful tool to create various controlled page designs for typographic study.

David Walden
My Boston: Some printing and publishing history

Boston, where I have lived for nearly 50 years, has an important publishing and printing history. Therefore, for TUG 2012 I have used various library and other Boston resources to learn more about the printing and publishing history of the city. This presentation sketches what I have learned about several eras of Boston printing and publishing: 1) Colonial period, 1630–1775; 2) Revolutionary War (1775–1783) and transition; 3) Literary culture of the mid-19th century; 4) Later 1800s to mid-1900s; 5) Personal observations, 1964–present.