TUG 2014 abstracts

Editor's note: Slides and other related information for many of the talks are posted at `http://tug.org/tug2014/program.html`.

– – ∗ – –

Kaveh Bazargan
Creating a LaTeX class file using a graphical interface
Writing LaTeX class files is a very specialized job, needing intimate knowledge of TeX. This job can be simplified by separating parameters from the TeX commands themselves. Using such parameterization, a user can simply change the parameters and obtain the change required. The technique becomes much more useful if a graphical user interface is used to modify the parameters interactively, and if the user gets instant feedback by seeing the result in the typeset document. I will demonstrate one such system, namely Batch Commander, created using LiveCode. I will show that many uncomplicated class files can be created by a user with minimal knowledge of TeX.

Kaveh Bazargan
PDF files both to print and to read on screen
PDF started life as a reliable format for distributing a document for printing. In the electronic age when most documents are never printed, many people have predicted the demise of PDF, in favour of other formats such as EPUB. But PDF has remained resilient and is likely to remain so for the foreseeable future. However, in general, a print PDF is not ideal for reading on screen, and vice versa. So publishers either compromise, by putting minimal links and enhancements, or by producing two completely separate PDFs.

I will present methods by which we can use TeX to create PDF files that print perfectly, but that are enhanced for online reading too.

Karl Berry
Building TeX Live
An overview of how TeX Live is constructed and updated, both so-called packages and programs. Here, packages contain only material which is interpreted, such as LaTeX add-ons, fonts, and scripts; these are updated throughout the year, as updates are released to CTAN. In contrast, programs are compiled binaries and are, almost always, updated only for the annual release. A document with details on all these topics (and much more) is available at `http://tug.org/texlive/doc/tlbuild.html` (or `.pdf`).

Dave Crossland
Metapolator: Why Metafont is finally catching on
This presentation follows up on my paper, "Why didn't METAFONT catch on?" presented at TUG 2008 and in *TUGboat* 29:3. In March 2013, Dave contacted the developers of the Metaflop web application, and met with designer Simon Egli in New York. Simon proposed a radical idea to enable typeface designers to harness the power of METAFONT's parametric capabilities without requiring them to write any METAFONT code. After a year of prototyping, Dave and Simon secured financial support from the Google Fonts project to fully develop such a tool with a team of collaborators from around the world. They call it Metapolator.

Ward Cunningham
Another wiki? OMG why?
We will reflect on the first wiki, what problem it solved, and what problems it sidestepped. We'll acknowledge the most famous wiki, Wikipedia, and especially recognize what they found the need to change. We will then describe the technological and social opportunity for another wiki, again a service nobody asked for, but now built with technology twenty years further along in the evolution of the web. For this conference: `http://tug.fed.wiki.org`.

We've made wiki servers simpler by moving rendering and sharing into the browser. We've used federation to solve some problems that plague other implementations, and to open an innovation space unserved by web technology until now.

Finally we'll declare our love for those who write, even a little, for thoughtful writing inoculates us from the ravages of consumerism.

Paulo Ney de Souza
A call for standards on the way to internationalization of TeX
Even though TeX is able to write and process documents in hundreds of languages, its own internal organization regarding use of language is next to nil. On this talk we will show how the introduction of standards like ISO-639, ISO-15924, ISO-3166-1 and RFC-5646 are producing real cross-pollination among projects like Babel, Polyglossia, BibLaTeX and CSL and driving the internationalization of TeX further.

Michael Doob
Using animations within LaTeX documents
TeX has always been in its heart a program for creating beautiful books. As TeX matured, markup was simplified using LaTeX, packages for creating beautiful tables of contents and indexes appeared, and colour was added. Graphics packages were also added allowed beautiful illustrations, but the resulting output was in essence a book. This era of progress is disappearing.

We now read novels on our smart phones and mathematical papers on our tablets. The objects being viewed are less like traditional books and have potential to be much more exciting. Fortunately, TEX is flexible enough to adapt to the changing environment. As a step supporting this change, we survey the varied techniques available to create animations within LATEX documents.

David Farmer
Converting structured LATEX to other formats
I will describe a project which converts research papers and textbooks in mathematics from LATEX to HTML, providing an alternative to online PDF documents. This project specifically targets journal papers and books, as examples of structured documents. Making use of the underlying structure of the LATEX document, the output more closely preserves the reader's ability to visually scan the document's contents and seamlessly incorporates references and citations.

Doug McKenna
Li$\frac{b}{t}$erate $\frac{TEX}{C}$ Top part: `JSBox`
A work in progress, `JSBox` is a self-contained library — written in portable C — that instantiates sandboxable, TEX-language interpreters within the memory space of any C, Objective-C, or C++ 32- or 64-bit client program.

Built and documented anew, `JSBox` is faithful to the TEX language's primitives, syntax, typesetting algorithms, measurements, data structures, and speed. At the same time, it fixes — in an upwardly compatible manner — a variety of important problems with or lacunæ in the current TEX engine's implementation. These include integral support for 21-bit Unicode, namespaces, OpenType font tables and metrics, job-specific 8-bit to 21-bit Unicode mapping, run-time settable compatibility levels, full 32-bit fixed-point math, and more. Especially pertinent to interactive applications — such as an eBook reader — is that all of a document's pages can optionally be kept as TEX data structures in memory after a job is done, with direct random access of any requested page exported to the client program's screen without file I/O or DVI or PDF conversion if unneeded.

Tracing (notably including recursive expansion, the re-tracing of interrupted commands, alignments, math, etc.) and all error messages have been significantly improved over what TEX does. (A detailed article on `JSBox`'s tracing appears in this issue, pp. 157–167.)

The author will demo what `JSBox` can do now, and discuss what it could do in the future.

Doug McKenna
Li$\frac{b}{t}$erate $\frac{TEX}{C}$ Bottom part: `literac`
`literac` is a command-line program that converts source code written in C, C++, Objective-C, Swift, Go, or other languages that use C-style commenting syntax (i.e., `//` and `/* ... */`) into a LATEX document.

Computer code is typeset verbatim (with optional line numbers). Comments, on the other hand, are stripped of their delimiters, presented in different styles based on context, and merged into paragraphs if possible. A significant amount of attention is paid to ensuring that TEX does "the right thing" in numerous edge cases. Within comments, a few special commands support common typesetting tasks, including verbatim and auto-verbatim code quotes, macros, moving source material forward for typesetting later in the document, inserting arbitrary TEX code for math displays, tables, or footnotes, suppressing both code and comment lines from being typeset, and visual cues for dividing a large program in one source file into chapters, sections, subsections, a `README` file, etc. This fosters better documented C code without imposing an intermediate `CWEB` (`CWEAVE`/`CTANGLE`) step on the source code's developer/tester.

The single implementation file, `literac.c`, is documented in the literate style it implements. It typesets itself into a 200+ page document that is its own user manual, its own implementation, the explanation of the program's internal design, and an excellent test suite for itself. Its author built `literac` to typeset the 90,000+ lines of source code — half of them comments — of the `JSBox` library, as commented using `literac`'s rules and commands.

Frank Mittelbach
Regression testing LATEX packages with Lua
For many years, LATEX 2ε has used a custom Perl script to perform regression testing on a large number of test files to ensure that any changes to LATEX do not break anything. This tests have also been useful in highlighting changes in TEX Live. One of the first steps in the modern development of the LATEX3 code was to produce a similar system to ensure that the development process was as smooth as possible. This build system, which also handled documentation typesetting and CTAN archiving, was written with a Makefile system, and eventually a Windows batch script was written as well.

These days, all TEX distributions ship with LuaTEX, which includes a standalone Lua interpreter; for the first time, TEX users can write platform-independent scripts that run without needing to install any additional frameworks. (E.g., Perl under

Windows.) Joseph Wright, accordingly, has rewritten the LaTeX3 build scripts in Lua (uploaded to CTAN in June 2014) to avoid maintaining two separate systems. As an added bonus, this new system, named `l3build`, is flexible and extensible enough to be used for any TeX package, and we hope the community will now take advantage of having an easy-to-use regression test suite available.

In this presentation, we will discuss how we use the `l3build` system, how we hope others will use it, and why regression testing is so important.

Ross Moore
"Fake spaces" with pdfTeX — the best of both worlds
When Donald Knuth wrote TeX he chose to omit space characters from the output, but instead carefully position the start of each word and punctuation character. This was to be able to better handle the idea of full-justification, as done by clever typists on manual typewriters. TeX's visual output seems clearly superior because of this.

Nowadays, however, other word-processing and text-presentation software seems to have largely abandoned full justification. Instead, window sizes can be resized causing the text to reflow on-the-fly. The presence of a space character as a word delimiter is important for this to work properly.

With the 2014 version of TeX Live, new primitives such as `\pdffakespace` are included within pdfTeX that allow a "fake space" to be inserted into the PDF content stream, occurring between words and at the end of lines. This is done only at the final output, so it does not affect the high-quality positioning of words. Now when the textual content is extracted from the PDF, by Copy/Paste or other means, a space character is indeed included in the extracted content. This is a requirement to meet PDF/A archival standards.

The author will demonstrate examples of the use of this `\pdffakespace`, and the other new primitives that control when and where it is used (e.g., not needed in mathematical content) for producing PDF/A and "Tagged PDF" for both archivability and accessibility. Also to be shown is how a fake space allows extra material, such as the LaTeX source of inline or displayed mathematics, to be included invisibly within the PDF. With a Select/Copy/Paste of the mathematical expression, this included source coding comes along with the pasted text.

Dan Raies
LaTeX in the classroom
The LaTeX skill-set required of a student or a researcher is vastly different than that required of a teacher. As a result, when one writes a test or homework assignment it often happens that the LaTeX we know dictates the questions we can ask. In this talk we examine some LaTeX techniques that will allow us to improve the materials that we use in the classroom. These techniques include ways to organize documents, ways to write solutions, and ways to create diagrams. We will examine some basic LaTeX strategies as well as some particular packages that are of use for teachers.

Will Robertson and Frank Mittelbach
LaTeX3 and `expl3` in 2014: Recent developments
The `expl3` programming layer for LaTeX3 has stabilised and is now being used by many people "in the wild" and for many different package types. In this talk we'll discuss the emerging popularity of `expl3` and some thoughts we have for rounding out its feature set. One area of recent development is case-changing in the Unicode era; TeX's `\uppercase` and `\lowercase` don't fulfil our needs when case changing is language-dependent and in some cases no longer a one-to-one mapping. On the other hand, those primitives are used extensively for various tricks in TeX programming, and one of `expl3`'s philosophies is to avoid 'tricks', so we can try to do something about that too.

Etienne Tétreault-Pinard
Plotly: Collaborative, interactive, and online plotting with TeX
TeX was designed with the goal of allowing anyone to produce high-quality documents with minimal effort, and to provide a system compatible on all computers, now and in the future (A. Gaudeul, Do Open Source Developers Respond to Competition?: The (LA)TeX Case Study, Social Science Research Network, 2006).

Plotly applies the same core principles to graphics. Plotly lets users collaboratively make and share interactive graphics online using Python, MATLAB, R, Excel data and TeX (MathJax) for free. Additionally, Plotly allows users to easily embed and export graphics for publication, while backing up your graphics, data and revisions in the cloud. This tutorial outlines Plotly's features and demonstrates how using Plotly creates unique workflows with emphasis on collaboration and reproducibility. More information is at `bit.ly/1vdF6Kp`.

Alan Wetmore
A quarter century of naïve use and abuse of LaTeX
In this talk I will recount my introduction to and experiences with LaTeX. Throughout this time installing and maintaining TeX and friends has become ever so much easier while the capabilities have grown enormously. Along the way I learned about many subjects I hadn't even known existed, and experimented with many facets of TeX.