TUG 2014 — program and information

The **main conference** will take place in the Nortonia Room, lower level (adjacent to courtyard).
Sunday, July 27, 5–7 pm: **opening reception and registration**, in the Pearl Room, lower level.
Monday, July 28: **concurrent LaTeX workshop**, Cheryl Ponchin, boardroom (across from the Nortonia Room).
Monday, July 28: **post-session TeXShop workshop**, Herb Schulz, boardroom.

Monday July 28	8:00 am	*registration*	
	8:50 am	Robin Laakso, TUG	*Opening*
	9:00 am	Ross Moore, Macquarie Univ.	*"Fake spaces" with pdfTeX — the best of both worlds*
	9:35 am	Frank Mittelbach, LaTeX3 Project	*Regression testing LaTeX packages with Lua*
	10:10 am	Doug McKenna, Mathemaesthetics	*Li$\frac{b}{t}$erate $\frac{\text{TeX}}{C}$ Top part:* `JSBox`
	10:45 am	*break*	
	11:00 am	Doug McKenna	*Li$\frac{b}{t}$erate $\frac{\text{TeX}}{C}$ Bottom part:* `literac`
	11:35 am	Paulo Ney de Souza, Books in Bytes	*A call for standards on the way to internationalization of TeX*
	12:10 am	Robert Beezer, Univ. of Puget Sound	*MathBook XML*
	12:45 pm	*lunch*	
	1:45 pm	*group photo*	
	2:00 pm	David Farmer, Amer. Inst. of Math.	*Converting structured LaTeX to other formats*
	2:35 pm	William Hammond, San Diego, CA	*Will static CSS someday suffice for online rendering of profiled LaTeX?*
	3:10 pm	Keiichiro Shikano, Tokyo, Japan	*The easy way to define customized XML-to-LaTeX converters*
	3:45 pm	*break*	
	4:00 pm	Julian Gilbey, London, UK	*Creating mathematical jigsaw puzzles using TeX and friends*
	4:35 pm	Pavneet Arora, Bolton, Canada	*SUTRA — A workflow for representing signals*
	5:15 pm	q&a	
Tuesday July 29	8:55 am	*announcements*	
	9:00 am	Karl Berry, TUG	*Building TeX Live*
	9:35 am	Adam Maxwell, Port Angeles, WA	*TeX Live Utility: A slightly-shiny Mac interface for* `tlmgr`
	10:10 am	Richard Koch, Univ. of Oregon	*MacTeX design philosophy vs. TeXShop design philosophy*
	10:45 am	*break*	
	11:00 am	Dave Crossland, Metapolator	*Metapolator: Why Metafont is finally catching on*
	11:35 am	Michael Sharpe, UC San Diego	*Recent additions to TeX's font repertoire*
	12:10 am	Etienne Tétreault-Pinard, Plotly	*Plotly: Collaborative, interactive, and online plotting with TeX*
	12:45 pm	*lunch*	
	2:00 pm	Joseph Hogg, Los Angeles, CA	*Texinfo visits a garden*
	2:35 pm	SK Venkatesan and CV Rajagopal, TNQ Books and Journals	*TeX and copyediting*
	3:10 pm	Boris Veytsman, George Mason Univ.	*An output routine for an illustrated book*
	3:45 pm	*break*	
	4:00 pm	Kaveh Bazargan, River Valley Tech.	*Creating a LaTeX class file using a graphical interface*
	4:35 pm	Will Robertson and Frank Mittelbach, Univ. of Adelaide & LaTeX3	*LaTeX3 and* `expl3` *in 2014: Recent developments*
	5:15 pm	q&a	
Wednesday July 30	8:55 am	*announcements*	
	9:00 am	David Allen, Univ. of Kentucky	*Experiences with TikzDevice*
	9:35 am	Andrew Mertz, William Slough, and Nancy Van Cleave, E. Illinois Univ.	*Typesetting figures for computer science*
	10:10 am	Michael Doob, Univ. of Manitoba	*Using animations within LaTeX documents*
	10:45 am	*break*	
	11:00 am	Dan Raies, Univ. of Oregon	*LaTeX in the classroom*
	11:35 am	Jim Hefferon, Saint Michael's College	*Moving an online book to paper*
	12:10 pm	Kaveh Bazargan	*PDF files both to print and to read on screen*
	12:45 pm	*lunch*	
	2:00 pm	Leyla Akhmadeeva and Boris Veytsman, Bashkir State Med. Univ. & GMU	*Typography and readability: An experiment with post-stroke patients*
	2:35 pm	Alan Wetmore, US Army	*A quarter century of naïve use and abuse of LaTeX*
	3:10 pm	Ward Cunningham, Portland, OR	*Another wiki? OMG why?*
	3:45 pm	*break*	
	4:00 pm	Tracy Kidder, MA & ME	*Trying to write about technical topics*
	4:35 pm	David Walden, moderator	*Panel: TeX and the Wider Wilder World —* Hefferon, McKenna, Mittelbach, Rowley, Sharpe
	≈ 5:15 pm	*end*	
	6:30 pm	*banquet*	at Bluehour (`bluehouronline.com`), 409 SW 11th Ave.

Conference logistics

- **Conference location**: The Mark Spencer Hotel (`http://www.markspencer.com`) at 409 SW 11th Ave., Portland, Oregon, 97205.

  —*Sessions:* in the Nortonia Room, on the lower level, off the courtyard.

  —*Opening reception:* in the Pearl Room (adjacent to the Nortonia Room), lower level, Sunday evening, 5–7 pm. Hors d'oeuvres and nonalcoholic beverages will be served.

  —*Breaks and lunches:* in the lower-level courtyard, with tables set in the Pearl Room.

  —*Workshops:* both the all-day LaTeX and post-session TeXShop workshops on the first day will be in the boardroom, across the hall from the Nortonia Room.
- **Registration**: Available during the opening reception and before the conference begins, Monday morning from 8–8:45 am. Please check in at the registration table to pick up your name tag, conference booklet, and other items.
- **Internet access:** Complimentary wireless is available to all conference participants in guest rooms and public spaces. Be sure to ask for an access code upon check-in at the hotel. We will post access codes for the conference rooms.
- **Local information:** Maps, parking, nearby restaurants, and other information can be found on the hotel web site `http://www.markspencer.com/location.htm`. Dave Walden's extensive guide to visiting Portland and the surrounding area is at `http://dw2.tug.org/portland`.

Banquet

The conference banquet will be held at L'Heure Bleue (private dining area of Bluehour Restaurant, `http://www.bluehouronline.com`), starting at 6:30 pm on Wednesday, July 30. The restaurant is located a few blocks from the hotel, at 250 NW 13th Ave., Portland, 97209; we enter on Everett St. between 12th and 13th (map on web site), which is separate from the main restaurant entrance.

We will have a few door prizes as usual. In addition, we will hold a 32–128 second soapbox at the banquet, where anyone can speak for a minimum of 32 seconds and a maximum of 128 seconds:

- You can reminisce about Stanford, TeX, Knuth, or hold forth on something else: report a success, gripe about a problem, lament a failure, share an insight, ask a question, or explain a solution.
- No intros, no questions, no hacking on earlier speakers; just you, the mike, and the audience ...
- ... and a moderator with a timer who will cut you off when your time is up.
- No slides, overheads, whiteboards, blackboards, flipcharts, chalk, markers, or other props.
- Come prepared or make it up on the spur of the moment — no experience necessary.

Special guests

We are honored this year to have two special guests joining us.

Ward Cunningham has worked for and consulted to daring startups and huge corporations. He has served as CTO, Director, Fellow, Principal Engineer and Inventor. He is best known for creating wiki. He leads an open-source project rebuilding wiki to solve more complex sharing situations addressing some of society's toughest problems. Ward founded movements in object-oriented, agile software, extreme programming and pattern languages. Ward lives in Portland, Oregon and works for New Relic, Inc. His web site is `http://c2.com/~ward`.

Tracy Kidder is the author of *Soul of a New Machine*, which won the Pulitzer Prize and National Book Award, *Mountains Beyond Mountains*, and many other books. His most recent is *Good Prose*, written with his long-time editor, Richard Todd. He is a member of the Partners in Health board of trustees. His web site is `http://tracykidder.com`.

Leyla Akhmadeeva and Boris Veytsman
Typography and readability: An experiment with post-stroke patients

Typography for challenged audiences has unique problems. There is a large amount of research about reading by people with impaired vision. Since reading is a complex process, other impairments, for example, cognitive problems, may also influence it. Should a publisher of texts for this audience be aware of this? Which typographical devices must be used for these texts?

In our previous reports we showed that serifs do not influence the readability and understandability of texts by healthy students. In this report we study the readability and understandability of serif and sans-serif texts by post-stroke patients. We discuss the experimental setup and preliminary results.

David Allen
Experiences with TikzDevice

The following is a brief description of R and the tikzDevice package; this material is from `http://www.r-project.org`, slightly edited. My TUG presentation will provide more detail about R and tikzDevice. Mostly however, the presentation will consist of examples and demonstrations.

R is a language and environment for statistical computing and graphics. It is a GNU project. R provides a wide variety of statistical (linear and nonlinear modeling, classical statistical tests, time-series analysis, classification, clustering, ...) and graphical techniques, and is highly extensible. R is often the vehicle of choice for research in statistical methodology, and it provides an open source route to participation in that activity. One of R's strengths is the ease with which well-designed publication-quality plots can be produced, including mathematical symbols and formulas where needed. Great care has been taken over the defaults for the minor design choices in graphics, but the user retains full control. R is available as free software under the terms of the Free Software Foundation's GNU General Public License in source code form. It compiles and runs on a wide variety of Unix platforms and similar systems (including FreeBSD and GNU/Linux), Windows and Mac OS X.

The `tikzDevice` package provides a graphics output device for R that records plots in a LaTeX-friendly format. The device transforms plotting commands issued by R functions into LaTeX code blocks. When included in a paper typeset by LaTeX, these blocks are interpreted with the help of Ti*k*Z — a graphics package for TeX and friends originally written by Till Tantau. Using `tikzDevice`, the text of R plots can contain LaTeX commands such as mathematics. The device also allows arbitrary LaTeX code to be inserted into the output stream.

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.

Robert Beezer
MathBook XML

There is a need for a source format that explicitly captures the structure of a scholarly document (such as a research article or a textbook) and subsequently allows for automated conversion to the wide variety of output formats that are currently popular. MathBook XML is an XML application designed for this purpose. It is lightweight and practical, with an emphasis on being a tool that is simple for authors to use, especially those creating science and mathematics texts.

XSL transformations to LaTeX and HTML are under heavy development now. The HTML output employs MathJax to render LaTeX syntax for mathematics and `knowls` for fine subdivisions of the content and cross-referenced material. It is easy for an author to specify the content of embedded Sage cells, Geogebra demonstrations, or video. Conversions to other formats, such as Sage and SageMathCloud worksheets, iPython notebooks, and EPUB will be made as the project matures.

I will present an overview of the XML elements being built, the design decisions behind their creation, the output of several of the current conversions, and plans for continued development.

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.

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 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.

We've make 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.

Julian Gilbey

Creating mathematical jigsaw puzzles using TEX and friends

Hermitech Laboratory has created Formulator Tarsia, which is free Windows-based WYSIWYG software for authoring mathematical jigsaws. These are puzzles made out of triangles and squares with questions and answers written along the edges of the shapes; the aim is to match them up correctly. This talk describes a similar TEX- and Python-based system which I have created, which should be able to run on most operating systems. It makes use of a text-based YAML input file, and also allows for output to a Markdown file for further processing.

William Hammond

Will static CSS someday suffice for online rendering of profiled LATEX?

The appearance of MathJax about five years ago demonstrated that CSS manipulated heavily with JavaScript in a platform-dependent way was sufficient for online rendering of mathematics in HTML pages. The issue with MathJax is speed, not quality.

More recently CSS has become more powerful. Although quality remains an issue, CSS by itself has become at least adequate for fallback online presentation.

One may speculate that, as CSS continues to evolve, mere CSS may entirely suffice not only for HTML documents but also for the direct online rendering of profiled LaTeX documents when presented using XML syntax.

In this talk I will attempt to address (1) what the LaTeX community might hope to see in future development of CSS and (2) how CSS might be useful in the future of LaTeX itself.

More information on the use of CSS with mathematics is available at `http://www.albany.edu/~hammond/demos/purecss/`.

[1] William F. Hammond, "Dual presentation with math from one source using GELLMU", *TUGboat: The Communications of the TeX Users Group*, vol. 28 (2007), pp. 306–311; also available online at `http://tug.org/TUGboat/tb28-3/tb90hammond.pdf`. A video recording of the presentation at TUG 2007, July 2007, in San Diego is available at `http://www.river-valley.tv/conferences/tex/tug2007/`.

[2] William F. Hammond, "LaTeX profiles as objects in the category of markup languages", *TUGboat: The Communications of the TeX Users Group*, vol. 31 (2010), pp. 240–247; also available online at `http://tug.org/TUGboat/tb31-2/tb98hammond.pdf`. A video recording of the presentation at TUG 2010, June 2010, in San Francisco is available at `http://river-valley.tv/latex-profiles-as-objects-in-the-%e2%80%9ccategory%e2%80%9d-of-markup-languages/`.

[3] T. Atkins, fantasai, & Rossen Atanassov, ed., "CSS Flexible Box Layout Module Level 1", World Wide Web Consortium, last call working draft (work in progress), March 25, 2014, `http://www.w3.org/TR/2014/WD-css-flexbox-1-20140325/`, (latest: `http://www.w3.org/TR/css-flexbox-1/`).

Jim Hefferon
Moving an online book to paper

People often talk about moving paper works to online but I recently needed to do the opposite. I have a book, written in LaTeX, that I give away and was finally convinced to provide a paper version. I will discuss some differences in delivering work in the two formats that current authors can apply, and some things about today's landscape of independent printing.

Joseph Hogg
Texinfo visits a garden

GNU Texinfo provides a flexible, straightforward means of documenting a collection of about 400 plants growing in the Herb Garden at The Huntington located in San Marino, California (`www.huntington.org`). With more than 120 acres, the Huntington is known for its Botanical Gardens, Library, and Art Collection. The Herb Garden is one of the smaller gardens, about one-quarter acre in size, but one of the richest in plant density and diversity with 25 beds of plants for culinary, medicinal, perfume, dye and fiber, and other applications. Updated every six months, there is a list for each bed, an index, a taxonomic ranking, and web links that provide docents and visitors with information about the Garden

Texinfo has several friends that contribute to this project: Inkscape provides the images of plant beds and several shell scripts featuring `sed`, `awk`, and other Unix tools chime in. With more than 60 pages, we encourage docents to use the Plant List on their computers at home and smart phones in the Garden.

Richard Koch
MacTeX design philosophy vs. TeXShop design philosophy

The philosophy driving MacTeX and the philosophy driving TeXShop are completely different. What's the difference, and why?

MacTeX: We install the platform-independent version of TeX Live, identical to the TeX available on all other platforms, and not modified in any way for the Mac.

TeXShop: The front end to TeX has modern features Apple recently introduced and users expect, and makes no bones about compatibility with front ends on other platforms.

Details: Why was TeXShop written using Cocoa and does it matter? (You bet it matters.) Why was it converted to 64 bits and did that matter? (Sure does.) Does TeXShop support automatic saving, automatic reloading of documents when restarting, and the Retina display? Yes, yes, and yes. How hard were they to do? (They came for free because of the first items in this paragraph.)

Is TeXShop constructed using recent features of Objective C such as properties and automatic reference counting? (We had to be dragged there kicking and screaming, but yes, as of April, 2014.) Why do these matter?

Will TeXShop support features of the new system Apple might unveil at WWDC in June, 2014? (How would I know, and yes, of course.)

Adam Maxwell
TeX Live Utility: A slightly-shiny Mac interface for TeX Live Manager

TeX Live Utility is a Mac OS X graphical user interface for the TeX Live Manager command-line tool. I'll discuss the goals of the program, several usage examples, and some of the tricky issues in wrapping a tool in order to make it accessible to GUI users. Many of these issues (e.g., error handling and selection of features to expose) are not platform-specific, so could also be of interest to non-Mac users.

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 sandbox-able, 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 (including recursive expansion, re-tracing interrupted commands, alignments, math, etc.) and all error messages have been significantly improved over what TEX does. 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.

Andrew Mertz, William Slough, and Nancy Van Cleave

Typesetting figures for computer science

Presentation of computer science concepts often benefits from informative diagrams and figures. These include linked lists, trees, graphs, circuits, stacks, arrays, code listings and memory layout, for example. However, producing these types of diagrams can be challenging. Fortunately, a number of TEX packages can be used for this purpose. In this talk, we will illustrate the use of a few packages — such as tikz, bytefield, forest, drawstack, and listings — that are well-suited for constructing high quality figures for computer science.

Frank Mittelbach

Regression testing LATEX packages with Lua

For many years, LATEX2$_\varepsilon$ 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 handed documentation typesetting and CTAN archiving, was written with a Makefile system, and eventually a Windows batch script was written as well.

Nowadays, 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 are included within pdfTeX that allow a `\pdffakespace` 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.

Herbert Schulz
Workshop: TeXShop tips & tricks

An interactive workshop for users of TeXShop who want to get to know about some of the lesser-known but useful features of that front end to a TeX distribution on the Mac.

Michael Sharpe
Recent additions to TeX's font repertoire

We will examine some newer introductions and renovations of font families available to all users of LaTeX and its descendants. Most of these will be Roman text font families, but some will be of peripheral nature, like typewriter fonts. Part of the discussion will be the similarities and differences between different renditions of the great font families, like Bembo, Garamond, Baskerville, Times and Palatino, as well as more recent names such as Charter, Utopia and Libertine. I'll also take up issues of matching math fonts.

Keiichiro Shikano
The easy way to define customized XML-to-LaTeX converters

Although LaTeX has definitely been one of the best typesetting tools for decades, we often need some more rigid data formats as the sources of long and/or structured documents like books. Using a customized XML or any kind of markup language as the input, and then, before typesetting the text and applying your preferable styles with LaTeX, converting the source data with XSLT into a more standardized one like DocBook might be one of the more practical solutions. However, this would require rather specialized skills for both XSLT and the standard itself, as well as the markup language and the typesetting tool you use.

It would be nice to be able to generate a special XML-to-LaTeX converter as needed, just by defining some rules relating each XML element directly to a LaTeX command or environment. We propose a relatively straightforward and intuitive tool to do that. It is written in a dialect of Scheme programming languages, and works as a DSL (Domain Specific Language) to let you define the set of conversion rules as S-expressions, which have the advantage in handling the tree structure of an XML. In fact, we have already created and published dozens of books using this tool, and will show some in this presentation.

There have been some alternatives in this area. ConTeXt has native XML support. TeXML provides an intermediate format for serializing XML into

LaTeX. There also are niches for tools like `pandoc`. The approach we are proposing here will not obviate these tools, but could be a good option when you are thinking of XML as a source format for your document to be typeset using LaTeX.

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`.

SK Venkatesan and CV Rajagopal

TeX and copyediting

Copyediting of a manuscript involves bringing consistency at many levels, with many kinds of local and non-local changes.

The LaTeX macros of the proposed copyediting package offer an excellent way to create a markup that can string together all the types of changes that are made by a copyeditor in a consistent way. The English language also has certain localization requirements that could be handled through language switches in the spirit of Babel package. Localization can be achieved through macros such as `\vara{color}` that take care of variant spellings. We also propose further a family of macros for other copyediting requirements such as parenthetical commas, serial commas, Latin abbreviations, first-secondly-thirdly, ... This copyediting package will simplify the task of copyediting and bring a higher level of quality to the final output.

Boris Veytsman

An output routine for an illustrated book

Output routines involving illustrations ("floating bodies" in the LaTeX lingo) are the most complex part of TeX. For the most part, algorithms used in TeX LaTeX and ConTeXt the basic concept is the flow of text, occasionally interrupted by illustrations that can be placed anywhere close the the point they are mentioned. The story is told mainly by the text, and illustrations have the secondary role.

We discuss the different case of an *illustrated book*, where the main story is told by the illustrations and their interaction. The simplest examples of such books are art albums. Another (surprising) example is the *FAO Statistical Yearbook*, where the story is told primarily by maps, charts and tables, while text has the secondary role.

We describe a concept of a relatively simple output routine for such books and its implementation in LaTeX.

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.

MacTeX Design Philosophy vs TeXShop Design Philosophy

Richard Koch

I went to the Apple Developer Conference in May, 2000. Developers at this conference were supposed to receive the release version of OS X. In the keynote address, Steve Jobs announced that the new release would be renamed OS X Public Beta with a price reduced from $130 to a handling fee of $15. After the keynote, a knowledgable friend translated : "OS X has been delayed by a year."

As a sop to the audience, Apple held a software raffle during this conference, the only time I've heard of them doing so. Every developer got something, but it soon transpired that almost everybody got a schlocky piece of software on a CD, shrink wrapped against a flimsy piece of cardboard.

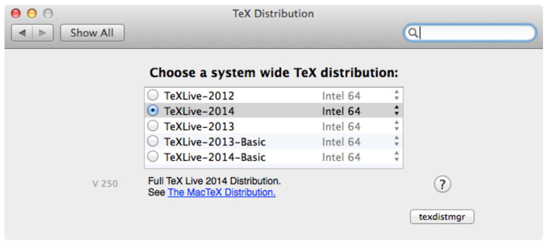
I was looking through this talk I agreed to give and it isn't very interesting. So I decided to give each attendee of the TUG conference a free piece of software.

The schlocky software Apple gave developers in 2000 was a forerunner of iTunes. This was before the iPod and all that. I, unfortunately, have nothing up my sleeve.

1 The Global PrefPane and the LocalTeX Pane

MacTeX installs a copy of TEX Live owned by root in `/usr/local/texlive`. It also installs a small data structure by Gerben Wierda and Jérôme Laurens in `/Library/TeX`, describing the distribution.These choices were somewhat controversial and I once gave a TUG talk about them. Now I'm happy.

Recall that each year's TEX Live distribution is in a folder named by date in `/usr/local/texlive`, so for instance TEX Live 2014 is in `/usr/local/texlive/2014`. This makes it possible to keep old distributions around, in case a new distribution breaks a crucial class file. We install a Preference Pane, shown below, for Apple's System Preferences, allowing users to switch between distributions. A switch changes all GUI apps to use the selected distribution and also changes the command line so command line programs use it.



Choose a system wide TeX distribution:

TeXLive-2012	Intel 64
[✓ TeXLive-2014]	Intel 64
TeXLive-2013	Intel 64
TeXLive-2013-Basic	Intel 64
TeXLive-2014-Basic	Intel 64

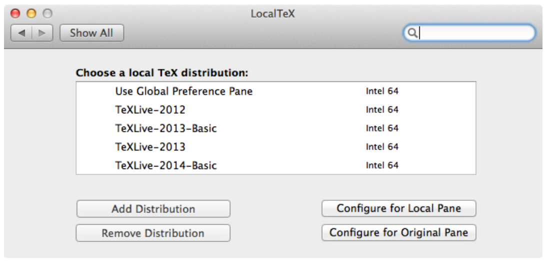
Full TeX Live 2014 Distribution.
See The MacTeX Distribution.

Figure 1: Global PrefPane

The PrefPane we install selects one distribution for all users and requires root access. I'm going to argue that we should have created a Local PrefPane instead, so each user could choose their own default TEX distribution and make this selection without root access. That's how *programs* work on the Macintosh. Programs live in `/Applications` where they are accessed by all users of a given machine. But each user has their own Preference settings for these applications, stored in `~/Library/Preferences`. One user's default Word font might be Times Roman, while another's might be Helvetica Neue.

The LocalTeX PrefPane shown on the next page is such a Pane. It can be installed locally for one user or globally for all users, but it makes independent choices for each user and does not require a password. This Pane does not change any link created by the Global Pref Pane or any element of the TeXDist structure, so it can be used together with the Global Pane, or when the Global Pane is completely missing.

The first item in the distribution list is always "Use Global Preference Pane." Selecting this item activates the Global Pane for the current user. The next items are distributions with TeXDist data structures, so an individual user can select a different default than the one chosen by the Global Pane.

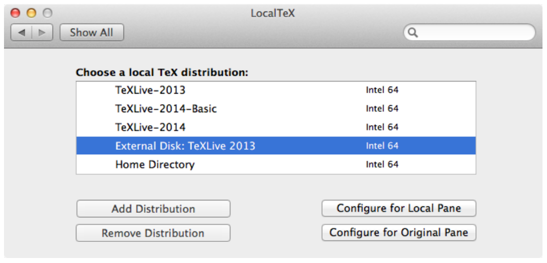


Choose a local TeX distribution:

Use Global Preference Pane	Intel 64
TeXLive-2012	Intel 64
TeXLive-2013-Basic	Intel 64
TeXLive-2013	Intel 64
TeXLive-2014-Basic	Intel 64

Figure 2: Local Pref Pane

Scrolling down in the list of distributions in the Pane, we see below that the LocalTeX pane can also define and select distributions on external disks, or distributions installed in a user's home directory. Although MacTeX cannot install TeX in such locations, the TeX Live install script from TUG can.



Choose a local TeX distribution:

TeXLive-2013	Intel 64
TeXLive-2014-Basic	Intel 64
TeXLive-2014	Intel 64
External Disk: TeXLive 2013	Intel 64
Home Directory	Intel 64

Figure 3: Local Pref Pane

Students may find this ability useful when they use a University owned machine and don't have root access. They can easily install TeX Live on a thumb drive, carry it with them, and have access to TeX in all locations.

The LocalTeX pane only shows distributions that are currently available. So if a thumb drive is removed, its distribution is no longer listed in the pane. Inserting the drive causes LocalTeX to list it again.

The "Add Distribution" button is used to inform the LocalTeX pane of TeX distributions without a TeXDist structure. It brings up a panel shown below. The "Name" field can be any desired name, since it will only appear in the LocalTeX pane. The "Path to Distribution" and "Path to Binaries" fields can be filled in by dragging appropriate locations to the dialog.

This data will only be accepted if the binary location is not empty, and contains a binary with at least one of the following names: tex, latex, pdftex, pdflatex, luatex, lualatex, xetex, xelatex.

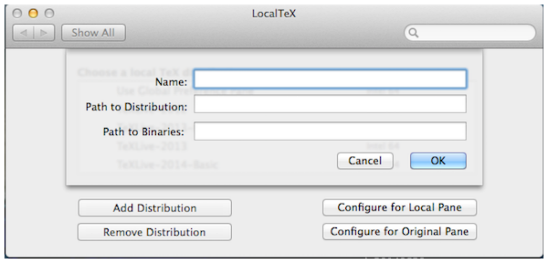


Figure 4: Local Pref Pane

The "Remove Distribution" button produces a list of extra distributions which can be removed one-by-one from those listed by the panel. Only distributions without a TeXDist data structure can be removed.

2 Installing and Configuring the LocalTeX Pane

The LocalTeX Pane can be obtained at `http://pages.uoregon.edu/koch/LocalTeX.zip`. Installing the LocalTeX pane is easy. Find and double click LocalTeX.prefPane. This brings up a dialog offering to install the Pane for all users or for only one user. Choose "only one user" and the Pane is installed for the current user without requiring a password. Or choose "all users" and the Pane is installed for everyone, but acts as a local pane for these users; installing this way requires a password.

After the Pane is installed, push the button "Configure for Local Pane" on the right. This reconfigures TeXShop, TeX Live Utility, and BibDesk to use the new Pane. It also reconfigures shell for *some* users, namely users whose home directory contains none of the three hidden files bash_profile, bash_login, and profile.

To return to the Global Pane and stop using LocalTeX, push "Configure for Original Pane" to reconfigure TeXShop, TeX Live Utility, and BibDesk. Don't do this if you are merely choosing "Use Global Preference Pane" in the LocalPane.

3 How Does the Pane Work?

The LocalTeX pane creates three symbolic links in `~/Library/TeX/LocalTeX`:

- texbin → binary directory of default distribution
- texroot → folder containing the default distribution
- texdist → texdist structure for the default distribution, if such a structure exists

GUI applications should be configured to look for TeX binaries in `~/Library/TeX/LocalTeX/texbin` rather than in `/usr/texbin`, the corresponding link for the Global pane. This is done automatically by the "Configure for LocalPane" button for TeXShop, TeX Live Utility, and BibDesk. LaTeXiT has a rather baroque preference system which doesn't permit setting its presences using the "defaults" command line tool, but they can be reset by hand, as can the corresponding preference settings for other third party applications. Many of these applications require a full path rather than one containing a tilde.

4 Other Advantages

Wierda and Laurens carefully selected the location for the link `/usr/texbin`, arguing that Apple would probably not change or remove this link. That reasoning turned out to be wrong, and users who upgrade OS X often find that they can no longer typeset even though their TeX distribution remains, because the link has been removed. The location `~/Library` does not present this problem because third party programs use it and wholesale Apple changes would create a nightmare.

Creating Preference Panes with root access requires dealing with Apple's security framework and that tends to change over time. The Local Pane is immune to security concerns. It currently runs on Yosemite betas. It requires Mountain Lion and above, since it uses Apple's newer ARC memory protection scheme.

5 Configuring Terminal

To finish installation of LocalTeX, add `/Users/koch/Library/TeX/LocalTeX/texbin` to your PATH before the item `/usr/texbin`. Here and in the following paragraphs, replace "koch" with your own login name.

If you use some other shell than the default bash, you no doubt know what to do already.

Otherwise follow these instructions. By default, modern versions of OS X use "bash" as a shall. This shell reads bash_profile when it starts up. If this file does not exist, it reads bash_login, and if this file does not exist, it reads profile. All three are hidden files, whose names start with a period.

Many users have none of these files. In that case, pushing the "Configure for Local Pane" button created a .bash_profile file for you and there is nothing more to do.

Otherwise, you need to edit the appropriate file. In Terminal, type

```
cd
ls -a
```

to see a list of hidden files in your home directory. If you have .bash_profile, edit that. If not, but you have .bash_login, edit that. Otherwise edit .profile. Make the same edit in all three cases. I'll discuss the case when you have .bash_profile. In Terminal, type

```
      cd
      mv .bash_profile bash_profile
```

Then you have a visible file to edit. Open this file with TeXShop. At the bottom, add the following two lines (the second and third lines below should be on a single line with no space between them).

```
# Added by LocalTeX Preference Pane
export PATH="/Users/koch/Library/TeX/
      LocalTeX/texbin":$PATH
```

and save the file. In Terminal type

```
      cd
      mv bash_profile .bash_profile
```

6 Removing Everything

If you install the LocalTeX Pane and decide that you don't want it, here is how to remove absolutely every trace from your computer.

- Using the Local Pane, push the "Configure for Original Pane" button to reconfigure TeXShop, TEX Live Utility, and BibDesk. If you reconfigured other apps, return them to their original configuration.
- Move LocalTeX.prefPane from `~/Library/PreferencePanes` to the trash.
- Move the folder LocalTeX from `~/Library/TeX` to the trash.
- If your shell was automatically configured for the new Pane, you will find a file named .bash_profile in your home directory containing the following lines. Throw the file in the trash.

  ```
  # Added by LocalTeX Preference Pane
  export PATH="/Users/koch/Library/
        TeX/LocalTeX/texbin":$PATH
  ```

  Otherwise you edited one of bash_profile, bash_login, or profile and added these lines. Remove them from the appropriate file.
- Finally, the LocalTeX Pref Pane stores its local data in the defaults system of OS X. To remove this data, type the following in Terminal (all three lines should be on a single line with spaces replacing line feeds):

  ```
  defaults remove
        com.apple.systempreferences
        localTeXExtrasData
  ```

7 LocalTeX and MacTeX

Will the LocalTeX preference pane be in a future edition of MacTeX? No. A choice between two Preference Panes would confuse most users, and while it is easy to configure the shell automatically for the global pane, this step requires user intervention for the local pane.

8 MacTeX Design Philosophy

From now on I'll give the promised talk. I work on the Macintosh in a small pond in the big TEX World. I wear two hats. I maintain MacTeX, the TEX install package for the Mac produced once a year by TUG. I also write, with collaborators, a GUI front end for TEX called TeXShop.

MacTeX is a "one button" package installing TEX , Ghostscript, and a few GUI applications. It presents a familiar interface for Mac users, asks no questions, and produces a completely configured installation. The installer was written by Jonathan Kew in an all night programming session at the North Carolina TUG Conference of 2005, and willed it to me at breakfast the next day. I was bleary eyed, but Jonathan was wide awake.

Jonathan's package installed a TEX distribution by Gerben Wierda, based on teTeX. But around this time, Thomas Esser abandoned teTeX and told his users to switch to TEX Live. Gerben produced a new distribution loosely based on TEX Live, which he announced at a TUG conference in Marrakesh in November of 2006. But at that same conference, he announced that he would immediately end support for the new distribution. This left us in a quandary and for several months it was unclear which distribution we would install. I had been attending TUG meetings since 2001, and in all that time, Karl Berry never asked me "why don't you Mac folks use TEX Live?" I thought TEX Live was nerdy and hard to install until I tried it and found installation very easy. Since 2007, we install a complete version of TEX Live.

Hence a "Design Philosophy for MacTeX." MacTeX installs *a completely unmodified full version of TEX Live on the Mac.* It is exactly the distribution used on Linux, Unix, and Windows (for those not using MikTeX). We refuse to reach into the distribution and make configuration changes. When someone complains "my Mac collaborators cannot typeset my code" we get to respond vigorously "Sir, it is YOUR fault because the Mac folks use standard TEX Live!"

Collaboration is common in research. Kunth worked very hard to make TEX produce the same results on all platforms. We have a responsibility to make TEX platform-independent. Open source forever!

(But a small voice: we are in Portland, Oregon, the home of Textures. Barry Smith rewrote the Pascal compiler for TEX , and then rewrote TEX to produce absolutely precise synchronization between source and output, and to support direct use of Macintosh fonts. His code was commercial, not open source. Textures users remember it with great passion. Every philosophy has a "yes, but ...")

9 TeXShop Design Philosophy

Surprisingly, TeXShop has a very different design philosophy than the MacTeX design philosophy. I'll argue that a GUI front end to TEX should rigorously follow the design standards of the particular platform it supports and should use the latest technology on that platform. This is difficult to achieve if the app supports many platforms.

To understand why, consider the following three messages from the TEX on OS X mailing list:

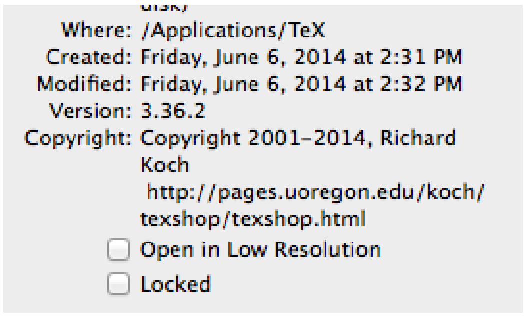
```
From: Warren Nagourney:
I am using TeXshop 2.47 on a retina MBP
and have noticed a slight tendency for
the letters in the preview window to be
slightly slanted from time to time.
The slant is enough to make the text
appear italicized, which is annoying.

From: Giovanni Dore:
I think that this is not a problem of
TeXShop. I use Skim and sometimes
I have the same problem.

From: Victor Ivrii:
Try to check if the same distortion
appears in TeXWorks and Adobe Reader:
TeXShop and Skim are PDFKit based,
while TW is poppler based and
AR has an Adobe engine.
```

All three messages are from knowledgable people active in the TEX on OS X list. As the third message states, TeXShop and Skim use Apple's PDFKit to display pdf files, while Adobe Acrobat Reader has its own pdf rendering code, and TeXWorks uses poppler to render pdf. And indeed, TeXShop and Skim have a display problem but Acrobat Reader and TeXWorks don't.

However, there is a missing ingredient here. The author of the original message has an Apple portable with a Retina Display. TeXShop and Skim support the Retina display because they were written with Apple's Cocoa language. Acrobat Reader and TeXworks don't support the Retina display, so Apple runs them in "magnify by two" mode. The real problem is a bug in Apple's Cocoa Retina code, subsequently fixed. The bug also goes away if you turn off Retina support in TeXShop and Skim.

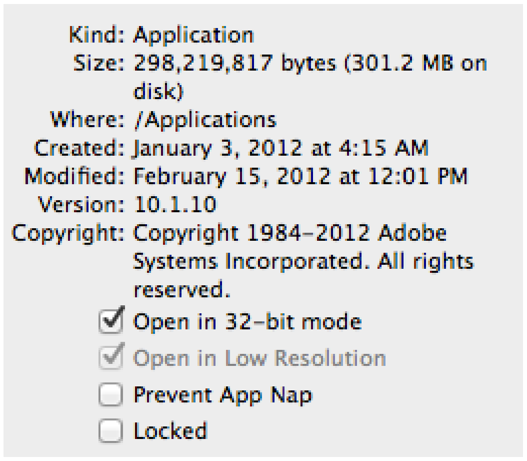
If you select "Get Info" in the Finder with a program selected, you get a panel of information about the program. Below is part of that panel for TeXShop on the top, and for Adobe Acrobat Reader on the bottom, displayed on a Retina machine.

The key difference is the option to open in Low Resolution mode. This is selectable in TeXShop but not in Reader. That means that TeXShop by default supports the Retina display, while Reader does not. In case of trouble, TeXShop can be converted to a mode in which it writes at normal resolution and the Mac magnifies by two, while Reader always runs in this magnify mode.



```
disk)
    Where: /Applications/TeX
  Created: Friday, June 6, 2014 at 2:31 PM
 Modified: Friday, June 6, 2014 at 2:32 PM
  Version: 3.36.2
Copyright: Copyright 2001-2014, Richard
           Koch
            http://pages.uoregon.edu/koch/
           texshop/texshop.html
           ☐ Open in Low Resolution
           ☐ Locked
```

Figure 5: About TeXShop



```
     Kind: Application
     Size: 298,219,817 bytes (301.2 MB on
           disk)
    Where: /Applications
  Created: January 3, 2012 at 4:15 AM
 Modified: February 15, 2012 at 12:01 PM
  Version: 10.1.10
Copyright: Copyright 1984-2012 Adobe
           Systems Incorporated. All rights
           reserved.
           ☑ Open in 32-bit mode
           ☑ Open in Low Resolution
           ☐ Prevent App Nap
           ☐ Locked
```

Figure 6: About Adobe Reader

The Retina Display Portable was introduced in June of 2012, but Adobe Reader and TeXWorks still don't support it two years later.

I had a very smart student who now works in the Portland software industry, so I boasted that TeXShop supported the Retina Display from the start. But he was too smart, and without skipping a beat he said "yeah, and how many lines of code did that take?" The answer is zero.

There are many ways to write GUI apps on the Mac: by supporting X11, by using Java, by using third party libraries, by using Carbon, and by using Cocoa. *If your app is written in Cocoa, then it automatically supports the Retina display. Otherwise not.*

10 NeXT at Apple, 1997 - 2007

Many of you read the book about Steve Jobs by Walter Isaacson. It is an interesting book, but has been criticized for getting the story of NeXT, and its role in Apple's second act, wrong. I agree, and here's a short version of that story from my perspective.

Apple bought NeXT in December of 1996, a sale that was finalized in February of 1997. Each May or June, Apple holds a Worldwide Developer Conference, WWDC. So in May of 1997, Apple had to give developers its strategy for using the NeXT operating system.

At the conference, Apple said that old Macintosh applications would continue to run in a sort of purgatory called the Blue Box, but new applications needed to be written in Objective C using NeXT's class library, then called OpenStep. Among commercial developers, the announcement went over like a lead balloon, and Apple got no significant endorsement at the conference. Apple's respected head of developer relations, Heidi Roizen, quit a few months later, calling the strategy "crazy."

So in 1998, Steve Jobs announced a completely different strategy. He called this new model

"Carbon" because, he said, "Carbon is the basis of all life." Carbon programs were written in C and C++ using the old Macintosh API, except that about 10% of the calls were replaced by new equivalents because the original calls wouldn't work on a modern multitasking operating system.This made it possible to start with an old Macintosh program, find the changed calls using an Apple-supplied script, revise them, and release the code on OS X. Apple immediately received endorsements from Microsoft, Adobe, Wolfram Research, and others, who stated that a whole new spirit of cooperation and realism was beginning to appear at Apple.

At this conference, OpenStep was renamed "Cocoa", but its standing at Apple was precarious. Some engineers said that new programs should be written in Cocoa, while others proclaimed vehemently that Cocoa was only for prototyping. At the 2000 developer conference I attended, the Carbon sessions were hald in the main auditorium packed with thousands of developers, while the Cocoa sessions were in a small church across the street, attended by 35 people who all seemed to know each other.

I attended WWDC regularly from 2003 to 2011, and this pattern continued for several years.

In 2005, Apple switched to Intel processors. At WWDC, they told developers that moving a Cocoa app to Intel usually involved a 10 minute recompile. Carbon apps, they estimated, could be moved in a month.

In 2006 the developer conference was postponed until August. At the conference, Apple gave developers a preliminary copy of Leopard, the next version of OS X, promising a release in March of 2007. A key feature of this release was full 64 bit support for all of Apple's important API's. Banners around the conference asked developers to become "64 bit ready" and a key slide of the keynote explained that "Leopard has full 64 bit support for Carbon and Cocoa."

But by June of 2007, Leopard was still not out. Why not? In January of that year, Apple announced the iPhone, and Apple engineers were pulled from the Leopard team to finish the software. But outside developers couldn't program the iPhone, so the 2007 conference was essentially a repeat of the 2006 version, with a keynote address using the same slides.

There was just one electric moment in 2007. Unfortunately, I completely missed its significance. When Jobs came to the slide promising "full 64 bit support for Carbon and Cocoa", the slide had been changed to read "full 64 bit support for Cocoa." Lots of developers noticed, and they mobbed Apple engineers during the lunch which followed the keynote. It rapidly became clear that Carbon was deprecated. Apple work on it had ceased.

So by 2007, Apple had the courage, and the prowess, to kill Carbon and throw their support totally behind Cocoa. Behind the scenes, they knew that both the iPhone and the as yet unannounced iPad could only be programmed in Cocoa. From 2008 on, there have been no Carbon sessions at WWDC. Commercial developers were among the last to switch to Cocoa, and some of their apps are still in Carbon.

During these turbulent times I was mostly oblivious to the drama. TeXShop remained a 32 bit app since I saw little reason to change.

But then TeXShop began crashing. I decided that the solution was to update to the latest Apple technologies. Shortly before Lion was announced, I moved TeXShop to 64 bits, and began planning to support garbage collection. What I didn't know was that dramatic changes were being made at Apple, and my 64 bit conversion was done just in the nick of time.

11 The Fragile Base Class Problem and 64 Bits

An *object* is a self-contained collection of code and date. Its data is referenced by variables known as *instance variables* and its code defines a series of *methods* or *functions.* According to a common metaphor, an object oriented program contains many objects , which talk to each other through method calls, and act on these calls by processing the data in their instance variables. Cocoa programs are object oriented.

To see how this works in practice, consider the Cocoa object called *NSView.* Each NSView corresponds to a rectangular portion of a particular window. The view has an instance variable pointing to it's window, a second instance variable giving the coordinates of its rectangular region, and so forth. Among the methods defined for an NSView are drawRect, which draws the view on the screen.

When a developer uses NSView, the developer defines a *subclass* of the view with a name like *myNSView.* This subclass has all the instance variables and methods of NSView, plus other instance variables and methods added by the programmer. But in addition, it can override some of the original methods of NSView. For instance, the drawRect command in NSView doesn't draw anything, but myNSView could override drawRect so it draws the logo of this conference. In this situation, we call NSView the *Base Class*, defined in Cocoa, and we call myNSView *a subclass* defined by the programmer.

The advantage of all this is that base classes usually come already connected up. Cocoa calls drawRect when the window first appears, when a covering window is moved out of the way, when a dialog box goes away, etc. Apple once gave developers a teeshirt with the text "Don't call us; we'll call you." The slogan means that the programmer's myNSView doesn't have to worry about when to draw because Cocoa will tell it when to draw. It just has to draw the logo when called.

The takeaway is easy: a Cocoa program runs cooperatively, with some tasks handled by the base classes in Cocoa and other tasks handled by subclasses defined by the programmer.

After object oriented programming appeared, programmers began to dream of a time when the system could be improved by just revising the base classes, without even recompiling the programs. You could install Mavericks, and suddenly say "wow, Word never did *that* before!"

Unfortunately, a barrier stood in the way of realizing this dream. The barrier was called "the fragile base class problem": *when revising base classes, you are not allowed to add extra instance variables or extra methods to the base class.* This was a problem in objective C, in C++, in Java, and elsewhere. The problem wasn't quite as bad in objective C as elsewhere, because it had been designed so extra methods in base classes are legal. But still: no extra instance variables.

When Apple added 64 bit libraries in the Leopard timeframe, they realized that they had a once in a lifetime opportunity to fix this problem. Since there were no existing 64 bit applications, every 64 app would have to be compiled from scratch. So they took the opportunity to make changes to objective C when run in 64 bits, including completely solving the fragile base class problem. If your Macintosh runs in 64 bits, then the dream of improving everything by revising the base classes can be realized.

Incidentally, they also made these changes in the iPhone even though it ran in 32 bits. So objective C on the iPhone, iPad, and 64 bit Mac applications is a different beast than objective C in 32 bit Mac applications.

After this change, Apple rapidly increased the hardware requirements of its operating systems. Snow Leopard required Intel processors, Lion required 64 bit processors, and Mountain Lion required machines running the kernel in 64 bits. After that the policy changed: Mavericks and Yosemite run on all machines that can run the previous systems. I believe that the reason for these policies is not that 64 bit programs run faster, but instead that Apple can now use all the extra added properties of objective C, including adding instance variables and methods to base classes.

12 Lion

Lion is the first Apple system to make real this great dream of improving programs by revising the base classes. Programs written in 64 bits with Cocoa got crucial added functionality for free, essentially without a recompile.

One of the standard requests for TeXShop was that it remember window sizes and positions when quit, and restore these windows automatically when next restarted. To shame me into working on this, users told me of other GUI's for TeX which already had the ability.

Imagine my surprise, then, when I discovered that TeXShop on Lion got the requested ability automatically for free.

An advantage of letting Apple do this is that Apple had second thoughts and slightly modified the behavior in Mountain Lion and Mavericks. TeXShop inherited those changes for free. For instance, it is now possible to turn the feature on or off in Apple's System Preferences. If windows are generally saved when quitting, then holding down the option key changes the Quit menu to "Quit and Close All Windows." If windows are not generally saved, hold the option key while quitting to save the windows. Finally, push the shift key when opening a program if you don't want to open old windows. These tricks work in TeXShop and all other Cocoa applications.

13 Automatic Saving

Saving window positions is something I could have done myself if I weren't lazy. But the second Lion feature is something I would never have tried on my own: automatic file saving.

Suppose you are using TeXShop in Lion, you have several source files open and have made changes in each. Suddenly you receive an emergency call and quit TeXShop. You won't receive pesky dialogs asking you to save each file; instead TeXShop will immediately quit.

But the next time you open TeXShop, your edit changes will be in all the source files.

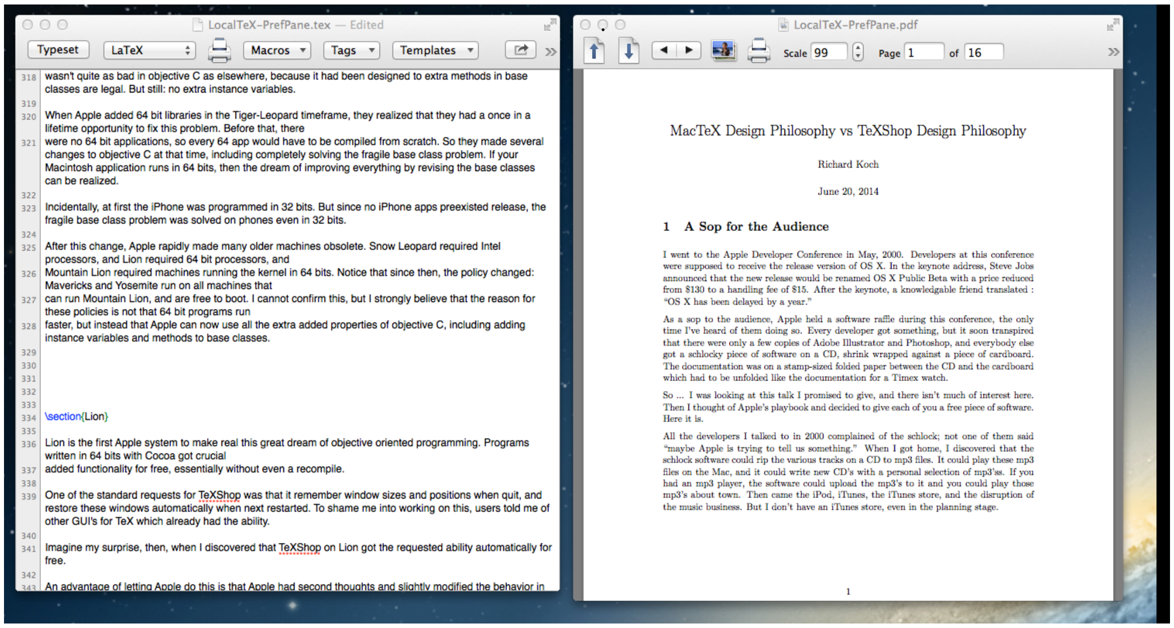
But wait — there's more. TeXShop doesn't just save when you quit. It saves every five minutes or so. If you live in a thunderstorm area with frequent power outages, no need to worry. When your computer starts up again, all that source you added will reappear.

"Gulp. Every five minutes the computer saves my 1000 page document?" Of course not. The program only saves changes, and in five minutes how much source did you change? In actual practice you never notice the saving process. There are no momentary glitches, no disk activity, and with a modern solid state drive no noise.

"Whoa. When I send a document to someone else, are all those changes in the document? My reference letter says 'works like a dog', but originally I wrote 'even a dog wouldn't be interested in his line of research.' " Not-to-worry, files only contain the latest version. That's done behind your back when you grab hold of a file to transfer it somewhere, and you won't notice.

"But there are so many edge cases where this scheme could go wrong." I absolutely agree. Indeed, I would never dare add automatic saving to TeXShop myself, or monkey around in any serious way with the file system. I dread getting a letter from a user claiming my program destroyed the only copy of his masterpiece. But this is Apple making the change, with a thousand engineers testing the code. Things slip by them, but destroying documents isn't something they'd take lightly.

▮



```
wasn't quite as bad in objective C as elsewhere, because it had been designed to extra methods in base
classes are legal. But still: no extra instance variables.

When Apple added 64 bit libraries in the Tiger-Leopard timeframe, they realized that they had a once in a
lifetime opportunity to fix this problem. Before that, there
were no 64 bit applications, so every 64 app would have to be compiled from scratch. So they made several
changes to objective C at that time, including completely solving the fragile base class problem. If your
Macintosh application runs in 64 bits, then the dream of improving everything by revising the base classes
can be realized.

Incidentally, at first the iPhone was programmed in 32 bits. But since no iPhone apps preexisted release, the
fragile base class problem was solved on phones even in 32 bits.

After this change, Apple rapidly made many older machines obsolete. Snow Leopard required Intel
processors, and Lion required 64 bit processors, and
Mountain Lion required machines running the kernel in 64 bits. Notice that since then, the policy changed:
Mavericks and Yosemite run on all machines that
can run Mountain Lion, and are free to boot. I cannot confirm this, but I strongly believe that the reason for
these policies is not that 64 bit programs run
faster, but instead that Apple can now use all the extra added properties of objective C, including adding
instance variables and methods to base classes.





\section{Lion}

Lion is the first Apple system to make real this great dream of objective oriented programming. Programs
written in 64 bits with Cocoa got crucial
added functionality for free, essentially without even a recompile.

One of the standard requests for TeXShop was that it remember window sizes and positions when quit, and
restore these windows automatically when next restarted. To shame me into working on this, users told me of
other GUI's for TeX which already had the ability.

Imagine my surprise, then, when I discovered that TeXShop on Lion got the requested ability automatically for
free.

An advantage of letting Apple do this is that Apple had second thoughts and slightly modified the behavior in
```

MacTeX Design Philosophy vs TeXShop Design Philosophy

Richard Koch

June 20, 2014

1 A Sop for the Audience

I went to the Apple Developer Conference in May, 2000. Developers at this conference were supposed to receive the release version of OS X. In the keynote address, Steve Jobs announced that the new release would be renamed OS X Public Beta with a price reduced from $130 to a handling fee of $15. After the keynote, a knowledgable friend translated : "OS X has been delayed by a year."

As a sop to the audience, Apple held a software raffle during this conference, the only time I've heard of them doing so. Every developer got something, but it soon transpired that there were only a few copies of Adobe Illustrator and Photoshop, and everybody else got a schlocky piece of software on a CD, shrink wrapped against a piece of cardboard. The documentation was on a stamp-sized folded paper between the CD and the cardboard which had to be unfolded like the documentation for a Timex watch.

So ... I was looking at this talk I promised to give, and there isn't much of interest here. Then I thought of Apple's playbook and decided to give each of you a free piece of software. Here it is.

All the developers I talked to in 2000 complained of the schlock; not one of them said "maybe Apple is trying to tell us something." When I got home, I discovered that the schlock software could rip the various tracks on a CD to mp3 files. It could play these mp3 files on the Mac, and it could write new CD's with a personal selection of mp3'ss. If you had an mp3 player, the software could upload the mp3's to it and you could play those mp3's about town. Then came the iPod, iTunes, the iTunes store, and the disruption of the music business. But I don't have an iTunes store, even in the planning stage.

1

Figure 7: Edit

Incidentally, there is still a "Save" button in case you've solved a great problem and absolutely want to write your discovery to disk.

"But wait. Suppose I delete some material, type an experimental new sentence, and then decide not to keep it. In the old system, I just don't save. But with automatic saving, the new stuff I don't want may be part of the document. Terrible!"

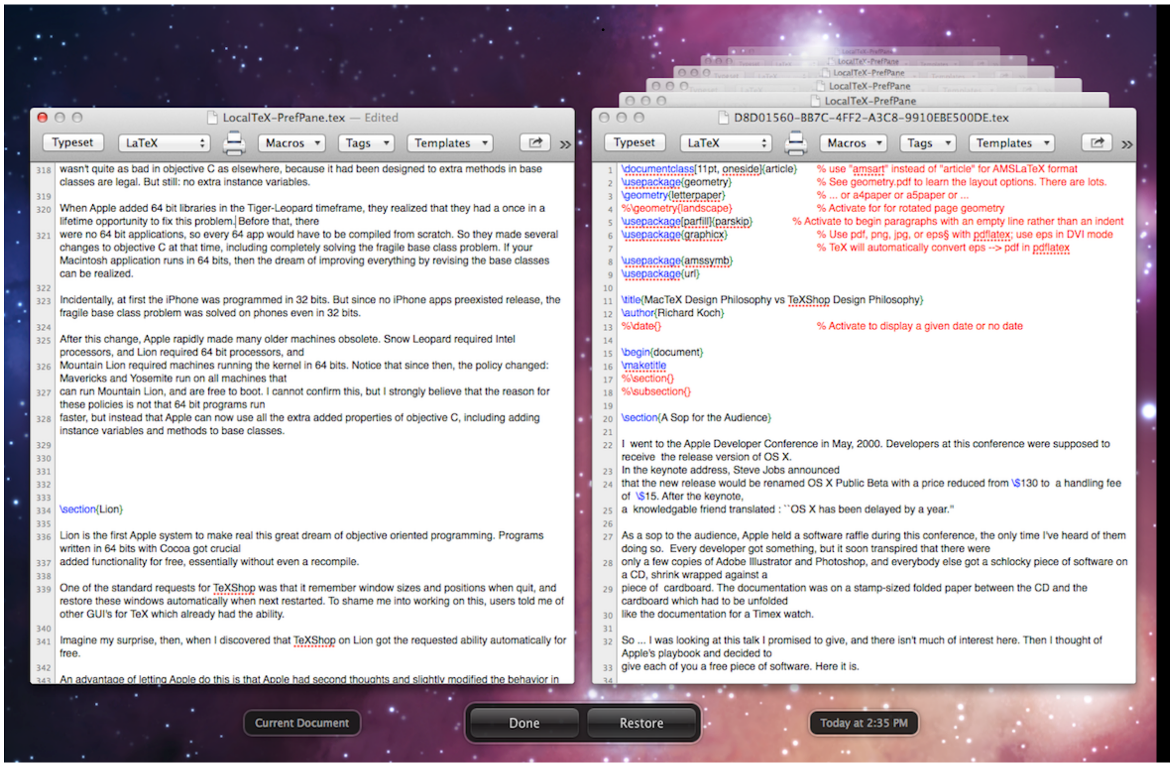
No, it's not. The top picture on this page shows the document you are reading while it was being edited. The top picture on the next page shows the effect of selecting the menu Revert To → Browse All Versions.

As you see, this gives a Time Machine view of the document, and we can retreat to an earlier version, or copy a portion of an earlier version to the current document. Time Machine need not be running to get this. Any application with AutoSave activated gets it for free.

I'll confess that I don't use Time Machine because I don't like the sound of a Disk Drive. The new feature gives Time Machine for TEX documents.

Apple has been refining the interface for AutoSave. It is intrusive on Lion, less intrustive on Mountain Lion, and less still on Mavericks. I couldn't live without it. If your TEX GUI has it, then it works the same as your other Mac applications.

AutoSave makes many changes under the hood. One of the most surprising is changes to program menus. The most controversial is the loss of a "Save As..." menu. I received many email messages demanding that I put back this menu. I replied that it was still present in my code, and Apple removed it while running the program. My correspondents found this explanation incomprehensible.



```
wasn't quite as bad in objective C as elsewhere, because it had been designed to extra methods in base
classes are legal. But still: no extra instance variables.

When Apple added 64 bit libraries in the Tiger-Leopard timeframe, they realized that they had a once in a
lifetime opportunity to fix this problem. Before that, there
were no 64 bit applications, so every 64 app would have to be compiled from scratch. So they made several
changes to objective C at that time, including completely solving the fragile base class problem. If your
Macintosh application runs in 64 bits, then the dream of improving everything by revising the base classes
can be realized.

Incidentally, at first the iPhone was programmed in 32 bits. But since no iPhone apps preexisted release, the
fragile base class problem was solved on phones even in 32 bits.

After this change, Apple rapidly made many older machines obsolete. Snow Leopard required Intel
processors, and Lion required 64 bit processors, and
Mountain Lion required machines running the kernel in 64 bits. Notice that since then, the policy changed:
Mavericks and Yosemite run on all machines that
can run Mountain Lion, and are free to boot. I cannot confirm this, but I strongly believe that the reason for
these policies is not that 64 bit programs run
faster, but instead that Apple can now use all the extra added properties of objective C, including adding
instance variables and methods to base classes.





\section{Lion}

Lion is the first Apple system to make real this great dream of objective oriented programming. Programs
written in 64 bits with Cocoa got crucial
added functionality for free, essentially without even a recompile.

One of the standard requests for TeXShop was that it remember window sizes and positions when quit, and
restore these windows automatically when next restarted. To shame me into working on this, users told me of
other GUI's for TeX which already had the ability.

Imagine my surprise, then, when I discovered that TeXShop on Lion got the requested ability automatically for
free.

An advantage of letting Apple do this is that Apple had second thoughts and slightly modified the behavior in
```

```
\documentclass[11pt, oneside]{article}      % use "amsart" instead of "article" for AMSLaTeX format
\usepackage{geometry}                       % See geometry.pdf to learn the layout options. There are lots.
\geometry{letterpaper}                      % ... or a4paper or a5paper or ...
%\geometry{landscape}                       % Activate for for rotated page geometry
\usepackage[parfill]{parskip}           % Activate to begin paragraphs with an empty line rather than an indent
\usepackage{graphicx}                       % Use pdf, png, jpg, or eps§ with pdflatex; use eps in DVI mode
                                            % TeX will automatically convert eps --> pdf in pdflatex
\usepackage{amssymb}
\usepackage{url}

\title{MacTeX Design Philosophy vs TeXShop Design Philosophy}
\author{Richard Koch}
%\date{}                                    % Activate to display a given date or no date

\begin{document}
\maketitle
%\section{}
%\subsection{}

\section{A Sop for the Audience}

I  went to the Apple Developer Conference in May, 2000. Developers at this conference were supposed to
receive  the release version of OS X.
In the keynote address, Steve Jobs announced
that the new release would be renamed OS X Public Beta with a price reduced from \$130 to  a handling fee
of  \$15. After the keynote,
a  knowledgable friend translated : ``OS X has been delayed by a year."

As a sop to the audience, Apple held a software raffle during this conference, the only time I've heard of them
doing so.  Every developer got something, but it soon transpired that there were
only a few copies of Adobe Illustrator and Photoshop, and everybody else got a schlocky piece of software on
a CD, shrink wrapped against a
piece of  cardboard. The documentation was on a stamp-sized folded paper between the CD and the
cardboard which had to be unfolded
like the documentation for a Timex watch.

So ... I was looking at this talk I promised to give, and there isn't much of interest here. Then I thought of
Apple's playbook and decided to
give each of you a free piece of software. Here it is.
```

Figure 8: Browse All Versions

The truth is that Apple automatically modifies the program Save menu when AutoSave is turned on. This is shown on the following page. On the left is the File menu as defined in current TeXShop source code. On the right is the actual menu as displayed in Mavericks. As can be seen, the middle section of the menu has been dramatically altered.

After one email exchange on "Save As", I wrote what I thought was a brilliant defense of Apple's actions, telling my readers to "grow up and go with the flow." The next day another user pointed out that "Save As..." had been restored by Apple in Mountain Lion. Sure enough, if you hold down the option key when accessing the File menu, "Duplicate" changes to "Save As." Apparently the people on the mailing list were also writing Apple.

The main point I'm trying to make here is that for programmers who use Cocoa, the solution of the Fragile Base Class Problem allows Apple to make surprisingly many changes under the surface.

After all this, you probably want me to come clean. To implement Auto Save, how much code did I actually write?

Apple's NSDocument object contains a function called autoSavesInPlace. This routine returns NO. In TeXShop I override it to instead return YES. That's it. One line of code gives AutoSave for free.

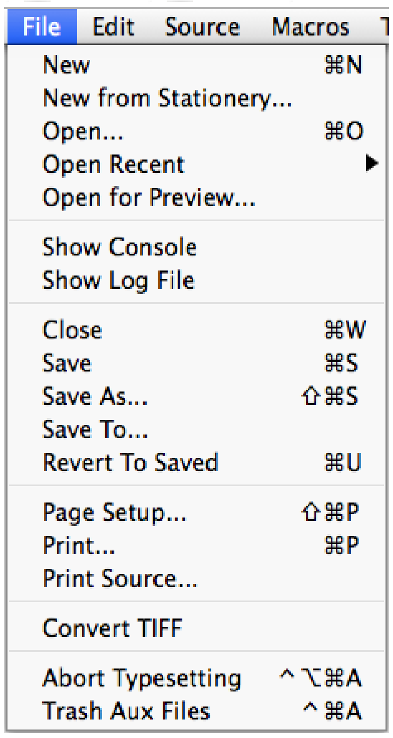


Figure 9: File Menu in Source Code

Incidentally, TeXShop's adoption of AutoSave has been popular. On the next page I show just one of the accolades I've received after releasing the Lion version.

Lots of collaborators help with TeXShop, providing features I haven't mentioned. Today I just wanted to show what is made possible by adhering to Apple's Cocoa standards.

TeXShop doesn't adopt everything, of course. It isn't in the Apple Store because working in a sandbox would limit its interaction with TEX Live and third party programs. It doesn't allow you to store documents in the Cloud because the Cloud is only available to applications in the store. But when an addition makes sense, it will be adopted.

So that's the end of my talk.....

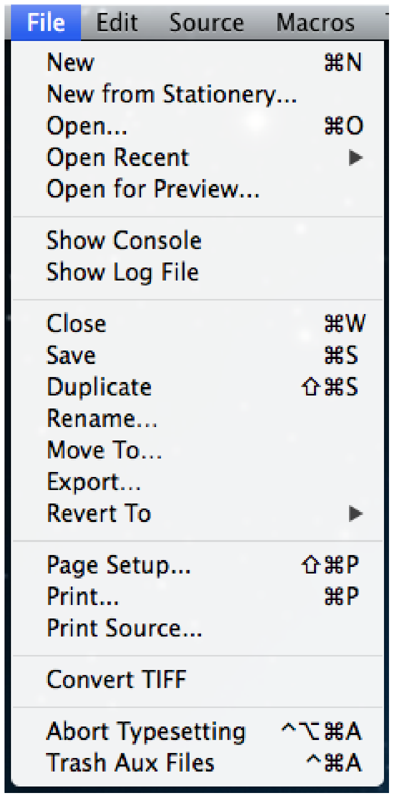


Figure 10: File Menu as Displayed by Mavericks

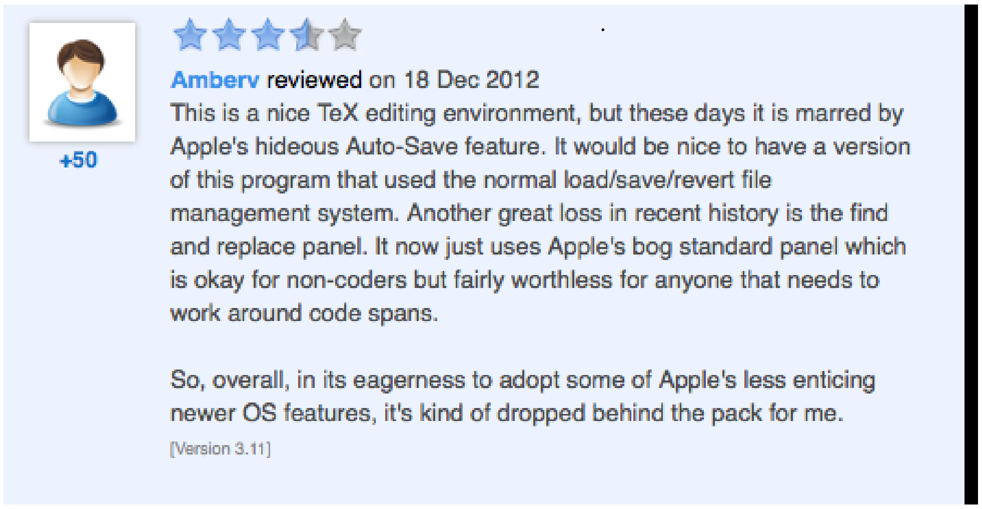
14 Automatic Reference Counting

But I hear one of you shouting me down.

"I couldn't care less about the Retina Display or Preserving Window Locations when Quitting, and I Hate Auto Save. Back there five or six pages, you mentioned "crashes". Why don't you talk about TeXShop crashes?"

OK.

One problem with object oriented programming is that a program can create hundreds of different objects as it runs. The program is supposed to throw away objects after it is done with them; if it doesn't, then computer memory becomes clogged and the program becomes sluggish.


Amberv reviewed on 18 Dec 2012

This is a nice TeX editing environment, but these days it is marred by Apple's hideous Auto-Save feature. It would be nice to have a version of this program that used the normal load/save/revert file management system. Another great loss in recent history is the find and replace panel. It now just uses Apple's bog standard panel which is okay for non-coders but fairly worthless for anyone that needs to work around code spans.

So, overall, in its eagerness to adopt some of Apple's less enticing newer OS features, it's kind of dropped behind the pack for me.

[Version 3.11]

Figure 11: A Review

Objects can be passed around, so just because one part of the program is done with an object doesn't mean that it isn't used by someone else. If an object is thrown away too soon, the program will crash when another part of the program tries to use the object.

There are three solutions. The first is to force programmers to manually handle memory management. That is how TeXShop worked until recently, and it is prone to errors that are hard to find.

The second method is called "garbage collection." Apple introduced it in Leopard, but it didn't work well on the iPhone.

Then as part of the enhancement of objective C, Apple introduced Automatic Reference Counting, or ARC, the third memory management technique. In ARC, the compiler automatically adds the code to handle memory management, and the programmer can ignore it. Since ARC does what a programmer would do managing memory manually, some files in a program can be compiled with ARC and some can be compiled without it.

This spring, I spent several weeks recompiling TeXShop with ARC, gradually working through the program file by file. The ARC code first appeared in TeXShop 3.34 and makes the program much more stable. A couple of remaining issues are solved in TeXShop 3.38, released at this conference, and this version ends the transition to ARC.

Adding ARC is an example of extensive work with no immediate gain; no interface changes are visible. It is essential work if the program is to survive for the long run.

Recent Additions to TeX's Font Repertoire

Michael Sharpe

Garalde Family

The first 150 years of the printing industry, beginning with Gutenberg in 1450, bear a striking resemblance to the early years of the personal computer industry. Both were intensely commercial enterprises, though with some high-toned gloss—Bibles then, scientific computing now. However, the real money driving the printers of the late 15th century was to a considerable extent *indulgences*—big money-makers for the Church as well as printers. As I learned from the fascinating books of Andrew Pettegree [2, 3], some monasteries were ordering from printers and selling to sinners hundreds of thousands of generic indulgences as soon as the technology to do so became available. The closest modern analogue may be the claim that pornographic movies drove the rapid growth of VCR and, later, DVD players.

The Lutheran Revolt of the early 16th century against the excesses of the Church did not hurt printers, as they worked overtime to print the voluminous tracts generated by the religious conflict. (One must bear in mind that the first newspaper did not appear until 1605.)

Given the importance of printed media in that period, it should not be surprising that much talent coalesced around the technology, and the fonts developed during that brilliant advance are, in my opinion, some of the most appealing ever created. They are referred to now as "old-style" or *Garalde* in honor of Aldus Manutius and Garamont.

Following Gutenberg, who worked with fonts we now call **Blackletter**, which remained the dominant ones in Germany through the first part of the 20th century, the first Roman font was developed by Nicolas Jenson of Venice, then the dominant commercial center of Europe, in the 1470's. Twenty years later, there appeared one of the great figures in publishing history—Aldus Manutius, also of Venice. Among other innovations, his company, the Aldine Press, invented the pocket book, italic type, greatly reduced the cost of books, standardized punctuation (introducing the semicolon), redefined book layout, and, through its "punchcutter" Francesco Griffo, whom we would now call a type designer, made a beautiful Roman font for the short book *De Aetna* by the poet Pietro Bembo, who became a major literary figure in the Italian Renaissance—lover of Lucrezia Borgia, major influence in standardizing the Italian language, creator of the *madrigal* form, and later, Cardinal of the Church. (The love letters between him and Lucrezia Borgia were considered by Lord Byron to be among the "prettiest" ever penned.) Modern renditions of the font used for *De Aetna* usually involve the name *Bembo*, though the basic free version is called *Cardo*, an obvious contraction of *Cardinal Bembo*. The fairly recent `fbb` package is based on Cardo, but with many changes—the ancient glyphs were stripped out, a kerning table was constructed for the Roman font, there being none in Cardo, and a Bold-Italic variant was created. Glyphs were added in all variants so that `fbb` has a full slate of `textcomp` characters and figures are available in proportional lining and oldstyle as well as tabular lining and oldstyle. Small Caps are provided in all variants. (Cardo had small caps only in Roman, regular weight.)

Sample showing `fbb`:

This is fbb, a free font package similar to Bembo. It has Small Caps, a very fine *Italic*, and a choice of number styles such as tabular oldstyle 0123456789.

Fifty years later, in Paris, Claude Garamont [Garamond] introduced and repeatedly refined his Roman and Italic fonts, based initially on the *De Aetna* font. Among the notable changes was the taming of *De Aetna*, reducing its ascenders and its over-arching "f", planing off some of its more prickly features and designing capital letters that looked less like the work of a scribe. The remarkable account of Garamont's fonts, their origins and influences, by Beatrice Warde [1] is highly recommended. The short version is that most Garamond fonts created in the early twentieth century were in fact based on later fonts by Jannon, not Garamont. Stempel Garamond (1925) is an exception, being based on a copy of the Egenolff-Berner specimen (see [1]) from 1592, owned by their foundry. The most recent Garamonds (URW++ Garamond No 8, Garamond Premier Pro, EBGaramond) have followed the same path. LaTeX now has a choice of two Garamonds:

- `garamondx` is an extension of Garamond No 8, adding small caps and oldstyle figures in both weights and both shapes. Because of the license, which is rather permissive but does not allow charging a fee, so cannot be distributed as part of TeXLive, though it can by MikTeX. Navigate to the URL `https://www.tug.org/fonts/getnonfreefonts` for a script you can download that will install garamondx on UNIX-like systems.
- `ebgaramond` (regular and italic only, no bold weights yet) is a very fine realization of Garamond that was recently added given TeX support.

Sample showing `ebgaramond`:

This is ebgaramond, a new realization of Garamond based on the Ebenolff-Berner specimen. It has very nice Small Caps, a very fine *Italic*, and a choice of number styles such as tabular oldstyle 0123456789.

Other Serifed Roman Families

Palatino:

Named for Italian writing master Giambattista Palatino, and inspired by Italian Renaissance fonts, Palatino has a larger xheight than typical old-style fonts and is more readable on-screen. It was one of the earliest fonts outside the Computer Modern family to gain TEX support, and remains of best-represented fonts for TEX.

- OpenType: TeX Gyre Pagella. Math available through Asana Math or TeX Gyre Pagella Math.
- PostScript: newpxtext + newpxmath, TeX Gyre Pagella + newpxmath, or mathpazo (text and math.) Can also use eulervm math for a more informal look.
- Kpfonts (complete text and math) are based on URW++ Palatino clone, but have their own distinctive, light appearance.

Times:

Many choices are now available.

- OpenType:
  - STIX (text + math);
  - TeX Gyre Termes + STIX math;
  - TeX Gyre Termes + TeX Gyre Termes Math;
- PostScript:
  - newtxtext + newtxmath/STIX;
  - TeX Gyre Termes + newtxmath/STIX;
  - STIX (text and math.)
- Mathtime (commercial but reasonably priced) is still a worthwhile Times-based math package, symbols lighter than STIX.
- Older choices such as mathptmx have now outlived their usefulness.

Baskerville:

A "transitional" font (c 1760), as was Plantin, the Times precursor. Baskerville ("the English Manutius"), was a master of fine detail, having been in the furniture finishing business (japanning) for a number of years. He set out to improve on Caslon, the then dominant font throughout England and its colonies. Baskerville's font was a favorite of Benjamin Franklin. Many commercial versions are available, most notably Storm Baskerville Pro. Free versions include:

- Baskervald (BaskervaldADF) was not designed with TEX in mind, and requires modifications to its ligature side bearings, its basic math character heights, and its kerning tables.
- (OpenType):
  Baskervaldx.otf, derived from BaskervaldADF, works OK with TEX.
- (PostScript):
  Baskervaldx + [baskervaldx]newtxmath works OK. Baskervald[x] lacks the high contrast that makes Baskerville stand out, and when scaled up to an xheight that matches the italic, it becomes a rather heavy Roman font.
- GFSBaskerville—for Greek, not Roman use.
- LibreBaskerville—lacks Bold Italic, and is designed as a web font, with larger xheight, larger counters and wider spacing than fonts intended for print output.

Sample showing `Baskervaldx`:

This is Baskervaldx, a font similar to Baskerville. It has Small Caps, *Italic*, and a choice of number styles such as tabular oldstyle 0123456789.

Utopia:

Utopia's design goals seem to have been to avoid any trace of old-style influence, and in this it has been very successful. The font looks quite austere, with tightly packed letters and, in my opinion, overly small inter-word spacing.

Adobe donated the four basic PostScript fonts to the X Consortium in 1992, though the terms of the license were not clear. In 2006, it was rereleased to the TEX User Group under clarified terms which allow modification and redistribution provided no name trademarked by Adobe is used.

- Fourier (Utopia text, fourier math) will make use of full (expert, Adobe) Utopia, if available.
- MathDesign [utopia] (Utopia text, MathDesign math) can also use expert fonts from Adobe.
- The ADF Venturis fonts are based on Utopia.
- An extension of the (free, basic part of) Utopia by Andrey Panov, dubbed Heuristica (Evristika), is available now from CTAN, TeXLive and MikTeX along with LaTeX support files. It has added ligatures, oldstyle and superior figures and Roman small caps, which seem too light for my taste, and can be used with matching math via [utopia]newtxmath. (Fourier and MathDesign cannot currently use the Heuristica extensions, being tied to Adobe's organization of Utopia Expert.)
- The LATEX support files for Heuristica now contain an option to set the factor by which to

multiply the interword spacing, `\fontdimen2`. The default value is 1, and the value 1.2 is suggested as a starting point.

Sample showing `Heuristica`:

This is Heuristica, an extension of Utopia. It has Small Caps, *Italic*, and a choice of number styles such as tabular oldstyle 0123456789.

Charter:

Bitstream contributed their four basic Charter fonts to the X Consortium under a very liberal license, and have been available in TEX for many years. Their low contrasts, high x-heights and use of piecewise linear outlines where possible may make them interesting again as fonts that will render well on small devices and perhaps projected slides. (Its worth noting that their designer, Matthew Carter, provided Georgia for Microsoft. It is widely considered to be one of the clearest serifed fonts for viewing on screen, and bears a number of similarities to Charter, though the latter is heavier.)

The XCharter fonts add oldstyle figures (proportionally spaced only), superior figures and small caps in all styles.

Sample showing `XCharter`:

This is XCharter, an extension of Charter. It has Small Caps, *Italic*, and a choice of number styles such as proportional oldstyle 0123456789.

Typewriter Fonts

The `courier` font that has long been available on CTAN is too light and too spread out for any use I can imagine in TEX, except to generate examples of what not to use. There are now several choices that are more attractive than you might expect for a monospaced font. Most are not new, but have been renovated recently so may appear new to you.

Serifed Typewriter Fonts:

- The `zlmtt` package provides access to all features of TeX Gyre Latin Modern Typewriter, a very substantial extension of cmtt. Best suited to lighter Roman fonts, though it can be scaled to be a better match up for some heavier Roman faces. The fonts themselves have been described thoroughly by Will Robertson in [4]. Its Small Caps are I think unique for a Typewriter font. The font does have a bold variant, but the boldness is almost imperceptible. The individual pieces are inconvenient to access through the `lmodern` package.
  `A sample of text using lmtt and its` **`bold variant.`**
- The `newtxtt` package is built on an enhanced version of the typewriter fonts contained in the `txfonts` package, with the addition of several choices of forms for 'zero'. The fonts are of the same width as `cmtt`, but are heavier and taller, matching Times weight and size. The newest version of the package has an option to reduce the interword space, so that, while it is no longer monospaced, it looks better for blocks of text that do not need to be aligned letter by letter.
  `A sample of text using newtxtt and its` **`bold variant.`**

Sans Serif Typewriter Fonts:

Two good packages are now available:

- Inconsolata–zi4 is an extension of Karl Berry's Inconsolata package, offering regular and bold weights, a choice of styles for 'zero', 'l' and quotes. It is based on an extension of Raph Levien's fine *Inconsolata* fonts, which are not dissimilar to Microsoft's *Consolas*.
  `A sample of text using inconsolata and its` **`bold variant.`**
- The `beramono` package is based on Bitstream's Vera Sans Mono. All glyphs are unmistakable. It is available only in T1 and TS1 encodings. The more recent DejaVu Sans package is a further extension with many more encodings and accented glyphs.
  `A sample of text using beramono and its` **`bold variant.`**

Sans Serif Fonts

There are now several choices of (proportionally spaced) sans serif fonts available to TEX users, among the more recent being cabin (similar to Gill Sans), raleway and SourceSansPro. As fonts of this type are frequently made available in a multiplicity of weights, their support files can profit from use of the `mweights` package that allows you to choose which weight will be called "regular" and which will be "bold", independent of Roman and Typewriter choices.

If you make use of a sans serif font in text, you may find it problematic to distinguish it from a sans serif Typewriter face. (If you use sans serif only for headings, as in German typographic usage, and only for headings, this does not apply.) Note one peculiarity of cabin if you use it for person@gmail.com. That white-on-black @ is unfortunate and would benefit from an alternate, black-on-white, symbol.

References

[1] Beatrice Warde, (Paul Beaujon). The "Garamond" types, sixteenth and seventeenth century sources reconsidered, `http://www.garamond.culture.fr/kcfinder/`

`files/3_3_4_article_beatrice_warde.pdf`. *The Fleuron*, pages 131–179, 1926.

[2] Andrew Pettegree. *The Invention of News: How the World Came to Know About Itself*. Yale University Press, New Haven, CT, 2011.

[3] Andrew Pettegree. *The Book in the Renaissance*. Yale University Press, New Haven, CT, 2014.

[4] Will Robertson. An exploration of the Latin Modern fonts. *The PracTEX Journal*, (1), 2006.

Do you need on-site training for LaTeX?

Contact Cheryl Ponchin at

`cponchin@comcast.net`

Training will be customized for your company needs.

Any level, from Beginning to Advanced.