

Practical T_EX 2005 — program and information

Tuesday
June 14

	courses 9 am–4:30 pm	
	Peter Flynn	<i>Practical T_EX on the Web</i>
	Steve Peter	<i>Introduction to ConT_EXt</i>
	Cheryl Ponchin	<i>Beginning and Intermediate L^AT_EX</i>
8–9 am	<i>registration</i>	
10:30 am	<i>break</i>	
12:30 pm	<i>lunch</i> (until 1:30 pm)	
3 pm	<i>break</i>	
5–7 pm	<i>registration & reception</i>	

Wednesday
June 15

8–9 am	<i>registration</i>	
9 am	Karl Berry, T _E X Users Group	<i>Welcome</i>
9:15 am	Nelson Beebe, University of Utah	keynote address: <i>The design of T_EX and METAPOST: A retrospective</i>
10:15 am	<i>break</i>	
10:30 am	Peter Flom, NDRI	<i>A true beginner looks at L^AT_EX</i>
11 am	Anita Schwartz, University of Delaware	<i>The art of L^AT_EX problem solving</i>
11:45 am	Peter Flynn, Silmaril Consultants	<i>L^AT_EX on the web</i>
12:30 pm	<i>lunch</i>	
1:30 pm	David Ignat, IAEA	<i>Word to L^AT_EX for a large, multi-author scientific paper</i>
2 pm	Steve Grathwohl, Duke University Press	<i>ConT_EXt: Better living through setups</i>
2:30 pm	Ronald Fehd, CDC	<i>Indexing, MakeIndex, and SAS</i>
3 pm	<i>break</i>	
3:15 pm	Andrew Mertz & William Slough, Eastern Illinois University	<i>Beamer by example</i>
4 pm	q & a	moderator: <i>Lance Carnes</i>
4:30 pm	<i>Birds of a Feather</i> (see following page)	

Thursday
June 16

9 am	Eitan Gurari, Ohio State University	<i>MathML via T_EX4ht and other tools</i>
9:45 am	John Burt, Brandeis University	<i>Typesetting critical editions of poetry with poemscol</i>
10:30 am	<i>break</i>	
10:45 am	Joseph Hogg, Los Angeles	<i>T_EX takes a walk on the green side</i>
11:30 am	Klaus Höppner, DANTE e.V. & TUG	<i>Strategies for including graphics in L^AT_EX documents</i>
12:30 pm	<i>lunch</i>	
1:30 pm	Tristan Miller, DFKI	<i>HA-Prosper: Producing beautiful slides with L^AT_EX</i>
2:15 pm	David Allen, University of Kentucky	<i>Dynamic presentations using T_EXpower and PSTricks</i>
3 pm	<i>break</i>	
3:15 pm	Jonathan Kew, SIL International	<i>An introduction to XeT_EX</i>
4 pm	q & a	moderator: <i>Anita Schwartz</i>
4:30 pm	<i>TUG members meeting</i>	
7:30 pm	<i>banquet</i> (see following page)	

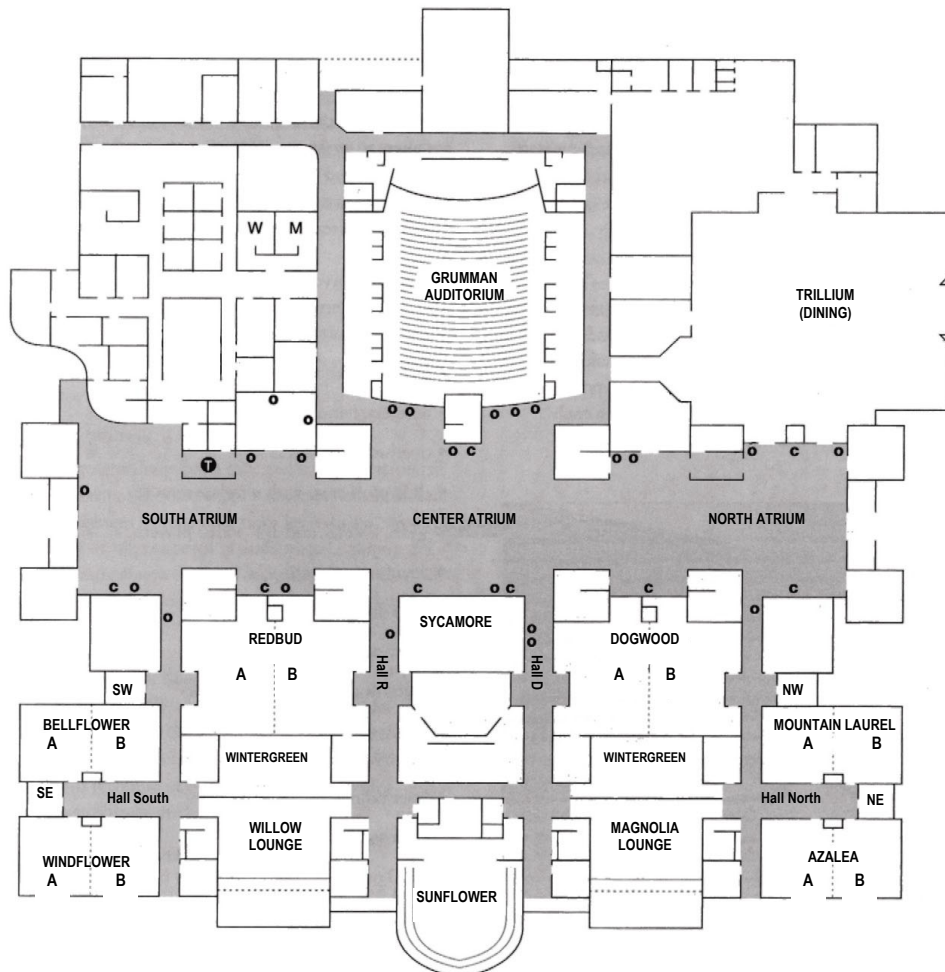
Friday
June 17

9 am	Tristan Miller	<i>Biblet: A portable BibT_EX bibliography style for generating highly customizable XHTML</i>
9:45 am	Volker R.W. Schaa, DANTE e.V.	<i>XML workflows and the EuroT_EX 2005 proceedings</i>
10:30 am	<i>break</i>	
10:45 am	Hans Hagen, Pragma ADE & NTG	<i>T_EX and XML</i>
11:30 am	Steve Peter	<i>T_EX font installation and usage</i>
12:30 pm	<i>lunch</i>	
1:30 pm	Mirko Janc, INFORMS	<i>L^AT_EX and PitStop: Unusual but powerful alliance</i>
2:15 pm	Kaveh Bazargan, River Valley Tech.	<i>A graphical user interface for T_EX</i>
3 pm	<i>break</i>	
3:15 pm	panel: Digital Publishing	moderator: <i>Steve Grathwohl; Kaveh Bazargan, Nelson Beebe, Lance Carnes, Peter Flynn, Hans Hagen, Mirko Janc.</i>
≈ 4 pm	<i>end</i>	

Conference logistics

All conference events (except the banquet) take place at the Friday Center for Continuing Education. A floor plan is included below.

- Registration is in Hall R, Tuesday morning 8–9 am, Tuesday evening 5 pm–5:30 pm, and Wednesday morning 8:30 am–9 am. Please come and pick up your name tag, conference information, and other goodies at one of these times if possible. Otherwise, see Robin Laakso.
- The reception is in Magnolia Lounge, Tuesday evening 5 pm–7 pm. It's an informal affair where you'll have a chance to nibble on Friday Center delicacies, get to know your fellow conference attendees, ask questions about what's coming up, brainstorm BOF ideas, and the like.
- Lunches will be served in the Trillium dining room.
- Breaks will be served in the Center Atrium. Food and beverages will also be available in the Atrium from 7:30 am–11 am each morning, and 1:30 pm–4:30 pm each afternoon (beverages from 1:30–4:30 and snacks from 2 pm–4 pm). Break food and beverages in the Atrium are shared with other conferences in the Friday Center, so feel free to help yourself to food and drink any time during these hours, regardless of TUG's scheduled breaks.
- The display area is in the Center Atrium.
- The main conference sessions are in Sunflower. *After hours use:* We can use Sunflower up to 7 pm, but must vacate promptly at 7.
- The classes on Tuesday will be held in Sunflower, Azalea A, and Azalea B.
- General wireless Internet access is available throughout the Friday Center. Also, the Center has four e-mail stations located in the Atrium.
- Free parking is available on the south side of the Friday Center; the second entrance on Friday Center Drive. No permit is required.
- *Higher Grounds*, a gift and book store managed by UNC Student Stores in conjunction with the Friday Center, is located just off the Atrium. Snacks, sundries, UNC gifts, and books are available.



Birds of a Feather

At this meeting we are reviving a long-standing tradition at TUG conferences, “Birds of a Feather” (BOF) meetings. The idea is to gather in informal groups to discuss a particular issue or topic(s). This will start Wednesday after the q & a.

One BOF session will be for Mac OS X and T_EX, and will take place in the main conference meeting room (Sunflower), since some remote participants are expected and thus there is extra setup to deal with. Other sessions can take place in the lounge, office, outside, or anywhere desirable.

Other BOF suggestions so far include ‘T_EX in educational environments’ and ‘Resources for T_EX beginners’. A list of these and any other suggestions we receive will be posted at the registration table along with signup sheets. If you would like to suggest additional topics at the conference, please let Robin Laakso know and we’ll add them to the list.

TUG members meeting

After the q & a on Thursday, we will hold a meeting for TUG members, and anyone else interested. Several TUG board members will be present at the conference: Karl Berry, Steve Grathwohl, Klaus Höppner, Steve Peter, and Cheryl Ponchin, as well as TUG’s executive director, Robin Laakso. We will report on TUG’s current status and future outlook.

More importantly, we invite discussion of any TUG-related business at this time: ideas for outreach to additional communities, ideas for additional initiatives TUG might undertake, existing projects which TUG might support, or anything else. Hope to see you there.

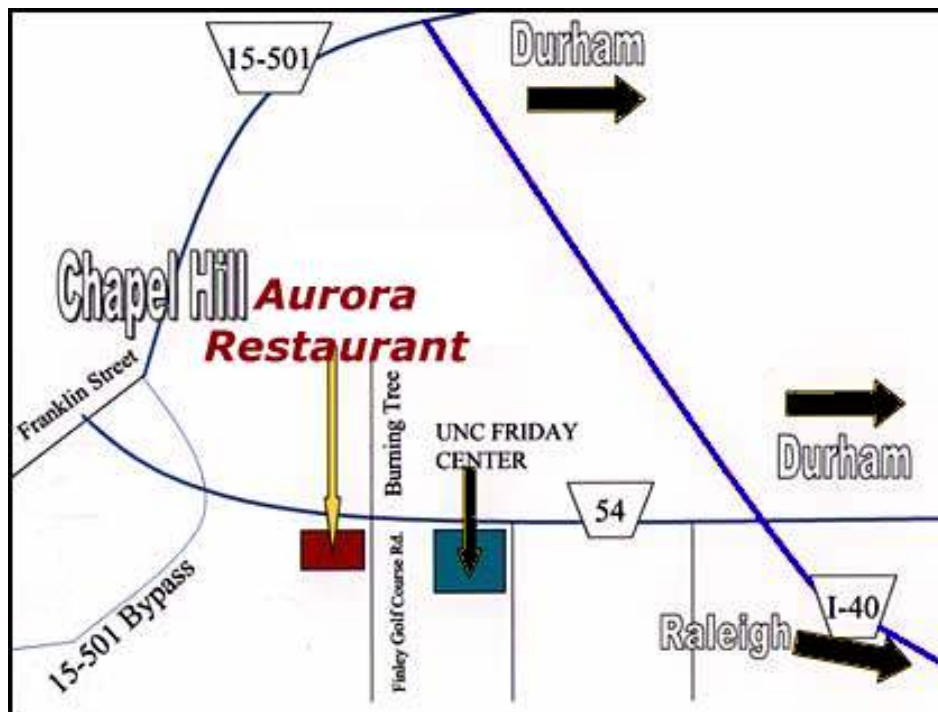
Banquet

At 7:30 pm Thursday, the conference will hold a banquet at the Aurora restaurant (1350 Raleigh Road, Chapel Hill, phone number 919-942-2400), one of the top Italian restaurants in the Triangle area for many years.

Directions: *If you’re driving from the Marriott*, take a left on Hwy 54, go two stop lights, and take a left at Friendly Forest Golf Course Road. Aurora Restaurant is a free standing building at the corner of Hwy 54 and Friendly Forest Golf Course Road.

If you’re walking from the Marriott, take the pathway adjacent to Hwy 54, on the same side of 54 as the Marriott, until you reach the restaurant. It’s about 10-minute walk.

Coming from other locations, please check Aurora’s on-line directions at <http://www.aurorarestaurant.com/directions.html>.



Dynamic presentations using \TeX power and PStricks

David Allen

A typical presentation consists of displaying a sequence of slides (a metaphor for screens) in a pre-determined order. This presentation is to demonstrate methods for preparing dynamic presentations in the following contexts:

1. Rather than showing a set of slides in pre-determined order, one may select the slides and their order after the presentation starts. This would likely be in response to questions from the audience.
2. If the discussion gets deep, it may be useful to visit a website or execute an external program.
3. A math professor might want to show a multi-line derivation one line at a time to focus attention to the current point of discussion.
4. An engineer might want to show a graphic depicting the assembling of a device one part at a time.

The \LaTeX packages used in this endeavor and their URLs follow.

Items 1. and 2. are implemented using the *hyperref* package, <http://www.ctan.org/tex-archive/macros/latex/contrib/hyperref>. Extensive facilities for navigation within a document, between documents, and with the web are provided by *hyperref*.

Items 3. and 4. are implemented using the *TeXpower* package, <http://texpower.sourceforge.net>. *TeXpower* is a \LaTeX package providing incremental display and special effects similar to those found in Microsoft PowerPoint.

Item 4. assumes there are graphics to be displayed, and my examples use graphics produced by the *PStricks* package, <http://www.pstricks.de>. *PStricks* provides a user friendly front end to the *PostScript* programming language. It is a generic \TeX package providing expansive computational graphics capabilities. These are the most novel examples.

A graphical user interface for \TeX

Kaveh Bazargan

I will demonstrate a graphical user interface that simplifies using global controls in \TeX . The software development environment I have used is Revolution (<http://www.runrev.com>). This is a successor to Apple's HyperCard, which is now unfortunately obsolete. One advantage of Revolution is that it is cross-platform, while HyperCard only ran on the Apple Macintosh.

The general idea is that the parameters and variables in \TeX (or \LaTeX or any other variant) are presented as interactive controls such as scrollbars and pop-up buttons in Revolution. As the user makes changes using these controls, the results are

immediately written to a configuration file and the main \TeX file is run and preview updated.

I will give a brief introduction to Revolution, and then demonstrate the \TeX GUI.

Typesetting critical editions of poetry with *poemscol*

John Burt

The *poemscol* package provides macros for \LaTeX for setting collections of poetry. It provides the structures required to produce a critical edition of the kind specified by the Modern Language Association's Committee on Scholarly Editions, providing line numbering, endnote sections for textual variants (both substantives and accidentals), emendations, and explanatory notes, and an index of titles and first lines. It provides running headers of the form "Emendations to pp. xx-yy" for the endnotes sections. It provides structures for different kinds of poetic text. It automatically marks every occasion where a stanza break falls on a page break. Aids for preparing parallel-text (as for instance editions with facing-page translations) editions are under development.

Indexing, MakeIndex, and SAS

Ronald Fehd

\LaTeX provides the *fancyvrb* package which is very useful in preparing a document which provides an overview of use of artificial-language programs used in data processing. This paper examines the theory of indexing and the \LaTeX *MakeIndex* package. The author provides two SAS programs which read all programs in a project directory and then write an index of intra- and inter-program references.

\LaTeX on the Web

Peter Flynn

Between \TeX users, distributing documents via the Internet (whether Web, email, or other facility) is easy because the types of files are known (*.tex*, *.dvi*) and they are easily reprocessed. Making quality typesetting available to others as single files is equally easy using PostScript or PDF. However, PostScript files can be huge, and they require a reader which few outside the graphic arts field have installed (although it's simple to do and freely available). PDF browsers are also readily available, as is HTML-style hyperlinking, but there are other limitations, including file sizes (especially with graphics), the lack of inward addressability, and the non-free nature of the file format itself.

Large or complex documents benefit from being split into chunks for serving, and from being served fast and light with HTML or XML and CSS. But HTML editors are notorious for their lack of structure and typographic or document-management facilities, and \LaTeX users are accustomed to having

these features at their disposal. Conversion from L^AT_EX to HTML is widely available, but unavoidably suffers from inherent the mismatch between feature-sets, and from the inherent reprogrammability of L^AT_EX. Authoring in XML, with conversion both to HTML/CSS and to PDF-via-L^AT_EX is one option, but has its own drawbacks in the learning curve and the early quality of some software.

This paper presents some techniques for use in authoring in L^AT_EX which can be used to minimize the conversion problems where a document is to be converted to HTML for serving to the Web, while continuing to produce the quality of typesetting for PS/PDF that we have become accustomed to.

MathML via T_EX4ht and other tools

Eitan Gurari

The support provided by graphical browsers for the HTML standard was a major ingredient in developing the Internet into a popular media for archiving and distributing general content. Two recent advancements suggest a similar bright future for mathematical content expressed through the MathML standard. The Mozilla Firefox browser, released last November, now offers native support for MathML. Also, the MathPlayer version 2 plug-in for MS Internet Explorer, which is easily installed and was released a year ago, is now capable of serving general MathML files.

This presentation will provide insight into how T_EX4ht produces MathML from L^AT_EX sources, and will consider issues involved in creating MathML with T_EX4ht and other tools.

T_EX takes a walk on the green side

Joseph Hogg

This talk describes three projects using various features of L^AT_EX and T_EX for non-profit and commercial projects.

1. *T_EX takes a walk on the green side*

The booklet “A Botanical Tour of the Los Angeles Zoo and Botanical Gardens” was typeset using L^AT_EX. This 36-page booklet plus cover, describes a self-guided tour of the LA Zoo’s botanical collection and includes a bloom calendar. The text was written by two docents, one of whom also created the botanical illustrations.

Using the report document class, the booklet features a table of contents with an integrated table of figures, drop caps, text-wrapping around figures, chapter quotes and an index with botanical and common names cross-referenced. Several packages were used to accomplish these effects: picins, quotchap, makeindx, letrine and caption being the major ones.

This project was done on a Windows 2000 system using the WinEdt editor. The discussion will

include sources of information, system platform, experimentation and workarounds for problems that came up during the project. The booklet was printed by a commercial printer from a pdf file generated by L^AT_EX.

2. *T_EX Meets a Maned Wolf at the Los Angeles Zoo*

L^AT_EX was used to create a leaflet describing the maned wolf, a handsome canid from South America. This leaflet was given to Zoo patrons during Wolf Awareness Week in October 2004. The leaflet document type was downloaded from TUG and the leaflet was created on a Macintosh system, then printed on a OKI C5150n color LED printer.

3. *T_EX Surveys the Local Real Estate Market*

L^AT_EX was used to typeset a 20-page booklet describing residential real estate values for the year 2004 in the Los Feliz, Silver Lake and Echo Park neighborhoods of Los Angeles. The article document class was used for text and photos and the PStricks package was used for tables and box-and-whisker plots. A pdf file was generated and the booklet’s pages imposed using Acro Software’s CutePDF. The booklet was printed on an OKI C5150n color LED printer. PStricks is an enjoyable package to use and I also typeset a card that can be used to send a short note to a client and present a business card.

L^AT_EX and PitStop: Unusual but powerful alliance

Mirko Janc

I will share some experiences in preparing art files for inclusion in L^AT_EX in the production cycle in our Institute. We publish 11 scholarly journals in Operations Research using L^AT_EX with a special font setup (presented at the TUG 2003 conference in Hawaii).

Powerful L^AT_EX math typesetting capabilities coupled with PitStop, a commercial Acrobat plugin, enable easy relabeling of figures with most complex math. Unlike other methods, exact positioning and scaling is a breeze. This same method we also use for updating colored covers where color issues are at stake, so the underlying PDF template can be properly preserved.

Some other related “tricks” to get clean art ready for proper inclusion in L^AT_EX will also be discussed.

An introduction to X_ET_EX

Jonathan Kew

Professor Donald Knuth’s T_EX is a typesetting system with a wide user community, and a range of supporting packages and enhancements available for many types of publishing work. However, it dates back to the 1980s and is tightly wedded to 8-bit

character data and custom-encoded fonts, making it difficult to configure $\text{T}_{\text{E}}\text{X}$ for many complex-script languages.

This paper will introduce $\text{XeT}_{\text{E}}\text{X}$, a system that extends $\text{T}_{\text{E}}\text{X}$ with direct support for modern OpenType and AAT fonts and the Unicode character set. This makes it possible to typeset almost any script and language with the same power and flexibility as $\text{T}_{\text{E}}\text{X}$ has traditionally offered in the 8-bit, simple-script world of European languages. $\text{XeT}_{\text{E}}\text{X}$ (currently available on Mac OS X, but possibly on other platforms in the future) integrates the $\text{T}_{\text{E}}\text{X}$ formatting engine with technologies from both the host operating system (Apple Type Services, CoreGraphics, QuickTime) and auxiliary libraries (ICU, TECKit), to provide a simple yet powerful system for multilingual and multiscript typesetting.

The most significant extensions $\text{XeT}_{\text{E}}\text{X}$ provides are its native support for the Unicode character set, replacing the myriad of 8-bit encodings traditionally used in $\text{T}_{\text{E}}\text{X}$ with a single standard for both input text encoding and font access; and an extended `\font` command that provides direct access by name to all the fonts installed in the user's computer. It also provides a mechanism to access many of the advanced layout features of modern fonts.

Additional features that will also be discussed include built-in support for a wide variety of graphic file formats, and an extended line-breaking mechanism that supports Asian languages such as Chinese or Thai that are written without word spaces.

Finally, we look briefly at some user-contributed packages that help integrate the features of $\text{XeT}_{\text{E}}\text{X}$ with the established $\text{L}^{\text{A}}\text{T}_{\text{E}}\text{X}$ system. Will Robertson's `fontspec.sty` provides a simple, consistent user interface in $\text{L}^{\text{A}}\text{T}_{\text{E}}\text{X}$ for loading both AAT and OpenType fonts, and accessing virtually all of the advanced features these fonts offer; Ross Moore's `xunicode.sty` is a package that allows legacy $\text{L}^{\text{A}}\text{T}_{\text{E}}\text{X}$ documents to be typeset using native OS X fonts without converting the input text entirely to Unicode, by supporting traditional $\text{T}_{\text{E}}\text{X}$ input conventions for accents and other 'special' (non-ASCII) characters.

Beamer by example

Andrew Mertz and William Slough

There are a variety of $\text{L}^{\text{A}}\text{T}_{\text{E}}\text{X}$ classes which can be used to produce "overhead slides" for presentations. One of these, `beamer`, provides flexible and powerful environments which can be used to create slides and PDF-based documents suitable for presentations. Although the class is extensively documented, first-time users may prefer learning about this class using a collection of graduated examples. The examples presented here cover a wide spectrum of use, from the simplest static slides to those with dynamic effects.

Biblet: A portable $\text{BibT}_{\text{E}}\text{X}$ bibliography style for generating highly customizable XHTML

Tristan Miller

We present Biblet, a set of $\text{BibT}_{\text{E}}\text{X}$ bibliography styles (`bst`) which generate XHTML from $\text{BibT}_{\text{E}}\text{X}$ databases. Unlike other $\text{BibT}_{\text{E}}\text{X}$ to XML/HTML converters, Biblet is written entirely in the native $\text{BibT}_{\text{E}}\text{X}$ style language and therefore works "out of the box" on any system that runs $\text{BibT}_{\text{E}}\text{X}$. Features include automatic conversion of $\text{L}^{\text{A}}\text{T}_{\text{E}}\text{X}$ symbols to HTML or Unicode entities; customizable graphical hyperlinks to PostScript, PDF, DVI, $\text{L}^{\text{A}}\text{T}_{\text{E}}\text{X}$, and HTML resources; support for nonstandard but common fields such as `day`, `isbn`, and `abstract`; hideable text blocks; and output of the original $\text{BibT}_{\text{E}}\text{X}$ entry for sharing citations. Biblet's highly structured XHTML output means that bibliography appearance can be drastically altered simply by specifying a Cascading Style Sheet (CSS), or easily postprocessed with third-party XML, HTML, or text processing tools. We compare and contrast Biblet to other common converters, describe basic usage of Biblet, give examples of how to produce custom-formatted bibliographies, and provide a basic overview of Biblet internals for those wishing to modify the style file itself.

$\text{T}_{\text{E}}\text{X}$ font installation and usage

Steve Peter

This talk is designed to be a near-comprehensive roadmap of installing and using fonts with $\text{T}_{\text{E}}\text{X}$ (with the exception of bitmapped fonts). We will start with the basics of $\text{T}_{\text{E}}\text{X}$ font handling (TFMs, etc.), along with a discussion of the major font technologies (PostScript, TrueType, and OpenType) and virtual fonts. Then we move to NFSS and `fontinst`, followed by `T_{\text{E}}\text{X}font` and `ConT_{\text{E}}\text{X}t` typescripts. Time permitting, we will configure an expert font, complete with `fi`, `fl`, `ff`, `ffi`, and `ffl` ligatures, suitable for professional typesetting.

The art of $\text{L}^{\text{A}}\text{T}_{\text{E}}\text{X}$ problem solving

Anita Schwartz

Have you ever been stuck using $\text{L}^{\text{A}}\text{T}_{\text{E}}\text{X}$? What does this really mean to you? "Stuck" may be anything from solving some esoteric error message while $\text{L}^{\text{A}}\text{T}_{\text{E}}\text{X}$ ing to trying to find a solution to a specific, not so obvious, formatting issue. There is a huge $\text{T}_{\text{E}}\text{X}$ community with a plethora of information where many problems have been solved by a highly knowledgeable group of volunteers. This presentation will attempt to lead you in the right direction to make the most out of the resources available during your $\text{L}^{\text{A}}\text{T}_{\text{E}}\text{X}$ adventure. I will attempt to explain common errors and provide solutions with $\text{L}^{\text{A}}\text{T}_{\text{E}}\text{X}$ and other variations such as `pdfL^{\text{A}}\text{T}_{\text{E}}\text{X}`.

Keynote Address: The design of T_EX and METAFONT: A retrospective

Nelson H. F. Beebe

University of Utah

Department of Mathematics, 110 LCB

155 S 1400 E RM 233

Salt Lake City, UT 84112-0090

USA

WWW URL: <http://www.math.utah.edu/~beebe>

Telephone: +1 801 581 5254

FAX: +1 801 581 4148

Internet: beebe@math.utah.edu, beebe@acm.org, beebe@computer.org

Abstract

This article looks back at the design of T_EX and METAFONT, and analyzes how they were affected by architectures, operating systems, programming languages, and resource limits of the computing world at the time of their creation by a remarkable programmer and human being, Donald E. Knuth. This paper is dedicated to him, with deep gratitude for the continued inspiration and learning that I've received from his software, his writing, and our occasional personal encounters over the last 25+ years.

1 Contents	
1 Contents	1001
2 Introduction	1001
3 Computers and people	1002
4 The DEC PDP-10	1002
5 Resource limits	1004
6 Choosing a programming language	1005
7 Switching programming languages	1006
8 Switching languages, again	1008
9 Wrapping up	1009

2 Introduction

More than a quarter century has elapsed since Donald Knuth took his sabbatical year of 1977–78 at Stanford University to tackle the problem of improving the quality of computer-based typesetting of his famous book series, *The Art of Computer Programming* [26, 27, 28, 29, 30, 31].

When the first volume appeared in 1968, most typesetting was still done by the hot lead process, and expert human typographers with decades of experience handled line breaking, page breaking, and page layout. By the mid 1970s, proprietary compu-

ter-based analog typesetters had entered the market, and in the view of Donald Knuth, had seriously degraded quality. When the first page proofs of part of the second edition of Volume 2 arrived, he was so disappointed that he wrote [35, p. 5]:

I didn't know what to do. I had spent 15 years writing those books, but if they were going to look awful I didn't want to write any more. How could I be proud of such a product?

A few months later, he learned of some new devices that used digital techniques to create letter images, and the close connection to the 0's and 1's of computer science led him to think about how he himself might design systems to place characters on a page, and draw the individual characters as a matrix of black and white dots. The sabbatical-year project produced working prototypes of two software programs for that purpose that were described in the book *T_EX and METAFONT: New Directions in Typesetting* [32].

The rest is of course history ... the digital typesetting project lasted about a decade, produced several more books [36, 37, 38, 39, 40, 34, 35], Ph.D. degrees for Frank Liang [44], John Hobby [16], Michael Plass [48], Lynn Ruggles [49], and Ignacio Zaballa Salelles [57], and had spinoffs in the commercial document-formatting industry and in the first laser printers. T_EX, and the L^AT_EX system built on top of it [9, 10, 11, 42, 43, 45], became the standard markup and typesetting system in the computer

science, mathematics, and physics communities, and has been widely used in many other fields.

The purpose of this article is to look back at \TeX and \METAFONT and examine how they were shaped by the attitudes and computing environment of the time.

3 Computers and people

Now that computers are widely available throughout much of the developed world, and when embedded systems are counted, are more numerous than humans, it is probably difficult for younger people to imagine a world without computers readily at hand. Yet not so long ago, this was not the case.

Until the desktop computers of the 1980s, a ‘computer’ usually meant a large expensive box, at least as long as an automobile, residing in a climate-controlled machine room with raised flooring, and fed electricity by power cables as thick as your wrist. At many universities, these systems had their own buildings, or at least entire building floors, called Computer Centers. The hardware usually cost hundreds of thousands to millions of dollars (where according to the US Consumer Price Index, a million dollars in 1968 is roughly the same as five million in 2000), and required a full-time professional staff of managers, systems programmers, and operators.

At most computer installations, the costs were passed on to users in the form of charges, such as the US\$1500 per hour for CPU time and US\$0.50 to open a file that I suffered with as a graduate student earning US\$1.50 per hour. At my site, there weren’t any disk storage charges, because it was forbidden to store files on disk: they had to reside either on punched cards, or magnetic tape. A couple of years ago, I came across a bill from the early 1980s for a 200MB disk: the device was the size of a washing machine, and cost US\$15,000. Today, that amount of storage is about fifty thousand times cheaper.

I have cited these costs to show that, until desktop computers became widespread, it was people who worked for computers, not the reverse. When a two-hour run cost as much as your year’s salary, you had to spend a lot of time thinking about your programs, instead of just running them to see if they worked.

When I came to Utah in 1978, the College of Science that I joined had just purchased a \DECSYSTEM 20 , a medium-sized timesharing computer based on the \DEC PDP-10 processor, and the Department of Computer Science bought one too on the same order. Ours ultimately cost about \$750,000, and supplied many of the computing needs of the College of Science for more than a dozen years, often sup-

porting 50–100 interactive login sessions. Its total physical memory was just over three megabytes, but we called it three quarters of a megaword. Although computer time was still a chargeable item, we managed to recover costs by getting each Department to contribute a yearly portion of the expenses as a flat fee, so most individual users didn’t worry about computer charges.

4 The DEC PDP-10

The PDP-10 ran at least eight or nine different operating systems:

- BBN TENEX,
- CompuServe 4S72,
- DEC TOPS-10 (sometimes jokingly called \BOTTOMS-10 by TOPS-20 users),
- DEC TOPS-20 (a modified TENEX affectionately called \TWENEX by some users),
- MIT ITS (Incompatible Time Sharing System),
- Stanford WAITS (Westcoast Alternative to ITS),
- Tymshare AUGUST, a modified TOPS-10, and
- Tymshare TYMCOM-X, and on the smaller \DECSYSTEM 20/20 model, TYMCOM-XX.

Although the operating systems differed, it was usually possible to move source-code programs among them with few if any changes, and some binaries compiled on TOPS-10 in 1975 still run just fine on TOPS-20 today.

Our machines at Utah both used TOPS-20, but Donald Knuth’s work on \TeX and \METAFONT was done on WAITS. That system was a research operating system, with frequent changes that resulted in bugs, causing many crashes and much downtime. Don told me earlier this year that the O/S was aptly named, since he wrote much of the draft of the \TeX book while he was waiting in the Computer Center for WAITS to come back up.

For about a decade, PDP-10 computers formed the backbone of the Arpanet, which began with just five nodes, at the University of California campuses at Berkeley, Los Angeles, and Santa Barbara, plus SRI (Stanford Research Institute) and Utah, and later evolved into the world-wide Internet [13, p. 48]. PDP-10 machines were adopted by major computer science departments, and hosted or contributed to many important developments, including at least these:

- Bob Metcalf’s *Ethernet* [Xerox PARC, Intel, and DEC];
- Vinton Cerf’s and Robert Kahn’s development of the *Transmission Control Protocol* and the *Internet Protocol* (TCP/IP);

- the MACSYMA [MIT], REDUCE [Utah] and MAPLE [Waterloo] symbolic-algebra languages;
- several dialects of LISP, including MACLISP [MIT] and PSL (Portable Standard Lisp) [Utah];
- the systems-programming language BLISS [DEC and Carnegie-Mellon University (CMU)];
- the shell-scripting language PCL (Programmable Command Language) [DEC and CMU];
- the SAIL (Stanford Artificial Intelligence Language) Algol-family programming language in which \TeX and METAFONT were first implemented;
- an excellent compiler for PASCAL [Hamburg/Rutgers/Sandia], the language in which \TeX and METAFONT were next implemented;
- Brian Reid’s document-formatting and bibliographic system, SCRIBE [CMU], that heavily influenced the design of $\mathbb{E}\TeX$ and $\text{BIB}\TeX$;
- Richard Stallman’s extensible and customizable text editor, EMACS [MIT];
- Jay Lepreau’s port, PCC20 [Utah], of Steve Johnson’s *Portable C Compiler*, PCC [Bell Labs];
- Kok Chen’s and Ken Harrenstien’s KCC20 native C compiler [SRI];
- Ralph Gorin’s SPELL, one of the first sophisticated interactive spelling checkers [Stanford];
- Mark Crispin’s mail client, MM, still one of the best around [Stanford];
- Frank da Cruz’s transport- and platform-independent interactive and scriptable communications software KERMIT [Columbia].

The PDP-10 and its operating systems is mentioned in about 170 of the now nearly 4000 *Request for Comments* (RFC) documents that informally define the protocols and behavior of the Internet.

The PDP-10 had compilers for ALGOL 60, BASIC, BLISS, C, COBOL 74, FORTH, FORTRAN 66, FORTRAN 77, LISP, PASCAL, SAIL, and SNOBOL, plus three assemblers called MACRO, MIDAS, and FAIL (fast one-pass assembler). A lot of programming was done in assembly code, including most of the operating systems. Indeed, the abstract of the FAIL manual [56] notes:

Although FAIL uses substantially more main memory than MACRO-10, it assembles typical programs about five times faster. FAIL assembles the entire Stanford time-sharing operating system (two million characters) in less than four minutes of CPU time on a KA-10 processor.

The KA-10 was one of the early PDP-10 models, so such performance was quite impressive. The high-level BLISS language might have been preferred for such work, but it was comparatively expensive to license, and few sites had it. Anyway, Ralph Gorin’s book on assembly language and systems programming [12] provided an outstanding resource for programmers.

Document formatting was provided by RUNOFF which shared a common ancestor ROFF with UNIX TROFF. Later, SCRIBE became available, but required an annual license fee, and ran only on the PDP-10, so it too had limited availability, and I refused to use it for that reason.

The PDP-10 had 36-bit words, with five seven-bit ASCII characters stored in each word. This left one bit, the low-order one, left over. It was normally zero, but when set to one, indicated that the preceding five characters were a line number that some editors used, and compilers could report in diagnostics.

Although seven-bit ASCII was the usual PDP-10 text representation, the hardware instruction set had general byte pointer instructions that could reference bytes of any size from 1 to 36 bits, and the KCC20 compiler provided easy access to them in C. For interfacing with 32-bit UNIX and VMS systems, 8-bit bytes were used, with four bits wasted at the low end of each word.

The PDP-10 filesystems recorded the byte count and byte size for every file, so in principle, text-processing software at least could have handled both 7-bit and 8-bit byte sizes. Indeed, Mark Crispin proposed that Unicode could be nicely handled in 9-bit UTF-9 and 18-bit UTF-18 encodings [6]. Alas, most PDP-10 systems were retired before this generality could be widely implemented.

One convenient feature of the PDP-10 operating systems was the ability to define *directory search paths* as values of *logical names*. For example, in TOPS-20, the command

```
@define TEXINPUTS: TEXINPUTS:,
ps:<jones.tex.inputs>
```

would add a user’s personal subdirectory to the end of the system-wide definition of the search path. A subsequent reference to `texinputs:myfile.tex` was all that it took to locate the file in the search path.

Since the directory search was handled inside the operating system, it was trivially available to all programs, no matter what language they were written in, unlike other operating systems where such searching has to be implemented by each program that requires it.

In addition, a manager could readily change the system-wide definition by a single privileged command:

```
^Edefine TEXINPUTS: ps:<tex.inputs>,
                ps:<tex.new>
```

The new definition was immediately available to all users, including those who had included the name `TEXINPUTS:` in their own search paths.

The great convenience of this facility encouraged those who ported \TeX and `METAFONT` to provide something similar. Today, users of the \TeX Live distributions are familiar with the `kpathsea` library, which provides an even more powerful mechanism for path searching.

The original PDP-10 instruction set had an 18-bit address field, giving a memory space of $2^{18} = 262\,144$ words, or about 1.25MB. Later designs extended the address space to 30 bits (5GB), but only 23 were ever implemented in DEC hardware, giving a practical limit of 40MB. That was still much more than most customers could afford in 1984 when the PDP-10 product line was terminated, and VAX VMS became the DEC flagship architecture and operating system.

DEC had products based on the KA-10, KI-10, and KL-10 versions of the PDP-10 processor. Later, other companies produced competing systems that ran one or more of the existing operating systems: Foonly (F1, F2, and F3), Systems Concepts (SC-40), Xerox PARC (MAXC) [8], and XKL Systems Corporation (TD-1, TOED-1, and TOAD-1). Some of these implemented up to 27 address bits (128MW, or 576MB). XKL even made a major porting effort of GNU and UNIX utilities, and got the X11 WINDOW SYSTEM running. Ultimately, none enjoyed continued commercial success.

The PDP-10 lives on among hobbyists, thanks to Ken Harrenstien’s superb KLH10 simulator [15] with full 30-bit addressing, and the vendor’s generosity in providing the operating system, compilers, and utilities for noncommercial use. On a fast modern desktop workstation, TOPS-20 runs several times faster than the original hardware ever did. It has been fun revisiting this environment that was such a leap forward from its predecessors, and I now generally have a TOPS-20 window or two open on my UNIX workstation.

5 Resource limits

The limited memory of the PDP-10 forced many economizations in the design of \TeX and `METAFONT`. Although `PASCAL` has `new()` and `dispose()` functions for allocating and freeing memory, imple-

Table 1: \TeX table sizes on TOPS-20 in 1984 and in \TeX Live on UNIX in 2004, as reported in the trip test.

Table	1984	2004	Growth
strings	1819	98002	53.9
string characters	9287	1221682	131.5
memory words	3001	1500022	499.8
control sequences	2100	60000	28.6
font info words	20000	1000000	50.0
fonts	75	2000	26.7
hyphen. exceptions	307	1000	3.3
stack positions (i)	200	5000	25.0
stack positions (n)	40	500	12.5
stack positions (p)	60	6000	100.0
stack positions (b)	500b	200000	400.0
stack positions (s)	600	40000	66.7

mentations were allowed to ignore the latter, so Don could not use them. Instead, all memory management is handled by the programs themselves, and sizes of internal tables are fixed at compile time. Table 1 shows the sizes of those tables, then and now. To further economize, many data structures were stored compactly with redundant information elided. Thus, for example, while \TeX fonts could have up to 256 characters, there are only 16 different widths and heights allowed, and one of those 16 is required to be zero. Also, although hundreds of text fonts are allowed, only 16 mathematical fonts are supported.

Instead of supporting scores of accented characters, \TeX expected to compose them dynamically from an accent positioned on a base letter. That in turn meant that words with accented letters could not be hyphenated automatically, an intolerable situation for many European languages. That restriction was finally removed in 1990 with the release of \TeX version 3.0 and `METAFONT` version 2.0, when those programs were extended to fully support 8-bit characters.

The \TeX DVI and `METAFONT` GF and TFM files were designed to be compact binary files that require special software tools to process. In contrast, in UNIX `TROFF`, these files are generally simple, albeit compact and cryptic, text files to facilitate use of filters in data-processing pipelines. Indeed, the UNIX approach of small-is-beautiful encouraged the use of separate tools for typesetting mathematics, pictures, and tables, instead of the monolithic approach that \TeX uses.

Finally, error diagnostics and error recovery reflect past technology and resource limits. Robin Fairbairns remarked in a May 2005 \TeX hax list posting:

Any \TeX -based errors are pretty ghastly. This is characteristic of the age in which it was developed, and of the fiendishly feeble machines we had to play with back then. But they're a lot better than the first Algol 68 compiler I played with, which had a single syntax diagnostic "*not a program!*"

6 Choosing a programming language

When Donald Knuth began to think about the problem of designing and implementing a typesetting system and a companion font-creation system, he was faced with the need to select a programming language for the task. We have already summarized what was available on the PDP-10.

COBOL was too horrid to contemplate: imagine writing code in a language with hundreds of reserved words, and such verbose syntax that a simple arithmetic operation and assignment `c = a*b` becomes

```
MULTIPLY A BY B GIVING C.
```

More complex expressions require every subexpression to be given a name and assigned to.

FORTRAN 66 was the only language with any hope of portability to many other systems. However, its lack of recursion, absence of data structures beyond arrays, lack of memory management, deficient control structures, record-oriented I/O, primitive Hollerith strings (`12HHELLO`, `WORLD`) that could be used only in `DATA` statements and as routine arguments, and its restriction to six-character variable names, made it distinctly unsuitable. Even so, it was later used elsewhere to implement a translation of METAFONT from SAIL for use on Harris computers [46].

PASCAL only became available on the PDP-10 in mid-1982, more than five years after Don began his sabbatical year. We shall return to it in Section 7.

BLISS was an expensive commercial product that was available only on DEC PDP-10, PDP-11, and later, VAX, computers. Although DEC later defined COMMON BLISS to be used across those very different 16-bit, 32-bit, and 36-bit systems, in practice, BLISS exposed too much of the underlying architecture.

LISP would have been attractive and powerful, and in retrospect, would have made \TeX and METAFONT far more extensible than they are, because any part of them could have been rewritten in LISP, and they would not have needed to have macro lan-

guages at all! Unfortunately, until the advent of COMMON LISP in 1984 [51, 52], and for some time after, the LISP world suffered from having about as many dialects as there were LISP programmers, making it impossible to select a language flavor that worked everywhere.

The only viable approach would have been to write a LISP compiler or interpreter, bringing one back to the original problem of picking a language to write *that* in. The one point in favor of this approach is that LISP is syntactically the simplest of all programming languages, so workable interpreters could be done in a few hundred lines, instead of the 10K to 100K lines that were needed for languages like PASCAL and FORTRAN. However, we have to remember that computer use cost a lot of money, and comparatively few people outside computer science departments had the luxury of ignoring the substantial run-time costs of interpreted languages. A typesetting system is expected to receive a lot of use, and efficiency and fast turnaround are essential.

PDP-10 assembly language had been used for many other programming projects, including the operating system and the three assemblers themselves. However, Don had worked on several different machines since 1959, and he knew that all computers eventually get replaced, often by new ones with radically-different instruction sets, operating systems, and programming languages. Thus, this avenue was not attractive either, since he had to be able to use his typesetting program for all of his future writing.

There was only one viable choice left, and that was SAIL. Although it had an offspring, MAINSAIL (Machine Independent SAIL), that might have been more attractive, that language was not born until 1979, two years after the sabbatical-year project. Figure 1 shows a small sample of SAIL, taken from the METAFONT source file `mfntpr.sai`. A detailed description of the language can be found in the first good book on computer graphics [47, Appendix IV].

The underscore operator in source-code assignments printed as a left arrow in the Stanford variant of ASCII (MIT also had its own flavor), but PDP-10 sites elsewhere just saw it as a plain underscore. However, its use as the assignment operator meant that it could not be used as an extended letter to make compound names more readable, as is now common in many other programming languages.

The left arrow in the Stanford variant of ASCII was not the only unusual character. Table 2 shows graphics assigned to the normally glyphless control characters. The existence of seven Greek letters in the control-character region may explain why \TeX 's

```

internal saf string array fname[0:2]
# file name, extension, and directory;

internal simp procedure scanfilename
# sets up fname[0:2];
begin integer j,c;
fname[0]_fname[1]_fname[2]_null;
j_0;
while curbuf and chartype[curbuf]=space
  do c_lop(curbuf);
loop   begin c_chartype[curbuf];
      case c of begin
        [pnt] j_1;
        [lbrack] j_2;
        [comma][wxy][rbrack][digit][letter];
      else done
      end;
      fname[j]_fname[j]&lop(curbuf);
    end;
end;

```

Figure 1: Filename scanning in SAIL, formatted as originally written by DEK, except for the movement of comments to separate lines. The square-bracketed names are symbolic integer constants declared earlier in the program.

default text-font layout packs Greek letters into the first ten slots.

Besides being a high-level language with good control and data structures, and recursion, SAIL had the advantage of having a good debugger. Symbolic debuggers are common today, sometimes even with fancy GUI front ends that some users like. In 1977, window systems had not yet made it out of Xerox PARC, and the few interactive debuggers available generally worked at the level of assembly language. Figure 2 shows a small example of a session with the low-level *Dynamic Debugging Tool/Technique*, DDT, that otherwise would have been necessary for debugging most programming languages other than SAIL (COBOL and FORTRAN, and later, PASCAL, also had source-level debuggers).

SAIL had a useful conditional compilation feature, allowing Don to write

```

# changed to ^P^Q when debugging METAFONT;
define DEBUGONLY = ^Pcomment^Q
...
# used when an array is believed to require
# no bounds checks;
define saf = ^Psafe^Q

```

Table 2: The Stanford extended ASCII character set. Character numbers are given in octal.

000	·	001	↓	002	α	003	β
004	∧	005	¬	006	ε	007	π
010	λ	011	γ	012	δ	013	∫
014	±	015	⊕	016	∞	017	∇
020	⊂	021	⊃	022	∩	023	∪
024	∀	025	∃	026	⊗	027	↔
030	_	031	→	032	~	033	≠
034	≤	035	≥	036	≡	037	∨
040–135 as in standard ASCII							
				136	↑	137	←
140–174 as in standard ASCII							
				175	◇	176	}
				177	^		

```

# used when SAIL can save time implementing
# this procedure;
define simp = ^Psimple^Q

```

```

# when debugging, belief turns to disbelief;
DEBUGONLY redefine saf = ^P^Q

```

```

# and simplicity dies too;
DEBUGONLY redefine simp = ^P^Q

```

A scan of the SAIL source code for METAFONT shows several other instances of how the implementation language and host computer affected the METAFONT code:

- 19 buffers for disk files;
- no more than 150 characters/line;
- initialization handled by a separate program module to save memory;
- bias of 4 added to case statement index to avoid illegal negative cases;
- character raster allocated dynamically to avoid 128K-word limit on core image;
- magic TENEX-dependent code to allocate buffers between the METAFONT code and the SAIL disk buffers because *there is all this nifty core sitting up in the high seg ... that is just begging to be used.*

7 Switching programming languages

Donald Knuth initially expected that \TeX and METAFONT would be useful primarily for his own books and papers, but other people were soon clamoring for access, and many of them did not have a PDP-10 computer to run them on. The American Mathematical Society was interested in evaluating \TeX and METAFONT for its own extensive mathematical publishing activities, but could make an investment in

```

@type hello.pas
program hello(output);
begin
  writeln('Hello, world')
end.

@load hello
PASCAL: HELLO
LINK:  Loading

@ddt
DDT
hello$b  hello+62$b  $$g
$1B>>HELLO/  TDZA 0  $x
  0/  0  0/  0
<SKIP>
HELLO+2/  MOVEM %CCLSW  $x
  0/  0  %CCLSW/  0
HELLO+3/  MOVE %CCLDN  $x
  0/  0  %CCLDN/  0
HELLO+4/  JUMPN HELLO+11  $x
  0/  0  HELLO+11
HELLO+5/  MOVEM 1,%RNNAM  $p

OUTPUT      : tty:
$2B>>HELLO+62/  JRST .MAIN.  $$x
Hello, world

```

Figure 2: Debugging a PASCAL program with DDT. The at signs are the default TOPS-20 command prompt. The dollar signs are the echo of ASCII ESCAPE characters. Breakpoints (\$b) are set at the start of the program, and just before the call to the runtime-library file initialization. Execution starts with \$\$g, proceeds after a breakpoint with \$p, steps single instructions with \$x, and steps until the next breakpoint with \$\$x.

switching from the proprietary commercial typesetting system that it was then using *only* if it could be satisfied with the quality, the longevity, and the portability of these new programs.

It was clear that keeping \TeX and METAFONT tied to SAIL and the PDP-10 would ultimately doom them to oblivion. It was also evident that some of the program-design decisions, and the early versions of the Computer Modern fonts, did not produce the high quality that their author demanded of himself. Researchers at Xerox PARC has translated the SAIL version of \TeX to MESA, but that language ran only on Xerox workstations, which, while full of great

ideas, were too expensive ever to make any significant market penetration.

A new implementation language was needed, and in December 1981, when the first source files for the new systems appeared, there was really only one possibility: PASCAL. However, before you rise to this provocation, why not C instead?

UNIX had reached the 16-bit DEC PDP-11 computers at the University of California at Berkeley in 1974. By 1977, researchers there had it running on the new 32-bit DEC VAX, but the C language in which much of UNIX is written was only rarely available outside that environment. Jay Lepreau’s PCC20 work was going on in the Computer Science Department at Utah in 1981–82, but it wasn’t until about 1983 that TOPS-20 users elsewhere began to get access to it. Our filesystem archives show my first major porting attempt of a C-language UNIX utility to TOPS-20 on 11 February 1983.

PASCAL, a descendant of ALGOL 60 [3], was designed by Niklaus Wirth at ETH in Zürich, Switzerland in 1968. His first attempt at writing a compiler for it in FORTRAN failed, but he then wrote a compiler for a subset of PASCAL in that subset, translated it by hand to assembly language, and was finally able to bootstrap the compiler by getting it to compile itself [54].

Urs Ammann later wrote a completely new compiler [1] in PASCAL for the PASCAL language on the 60-bit CDC 6600 at ETH, a machine class that I myself worked extensively and productively on for nearly four years. That compiler generated machine code directly, instead of producing assembly code, and ran faster, and produced faster code, than Wirth’s original bootstrap compiler. Ammann’s compiler was the parent of several others, including the one on the PDP-10.

PASCAL is a small language intended for teaching introductory computer programming skills, and Wirth’s book with the great title *Algorithms + Data Structures = Programs* [55] is a classic that is still worthy of being studied. However, PASCAL is *not* a language that is suitable for larger projects. A fragment of the language is shown in Figure 3, and much more can be seen in the source code for \TeX [37] and METAFONT [39].

PASCAL’s flaws are well chronicled in a famous article by Brian Kernighan [17, 18]. That paper was written to record that pain that PASCAL caused in implementing a moderate-sized, but influential, programming project [19]. He wrote in his article:

PASCAL, at least in its standard form, is just plain not suitable for serious programming. ... This botch [confusion of size and type]

```

PROCEDURE Scanfilename;
  LABEL 30;
BEGIN
  beginname;
  WHILE buffer[curinput.locfield] = 32 DO
    curinput.locfield := curinput.locfield+1;
  WHILE true DO
  BEGIN
    IF (buffer[curinput.locfield] = 59) OR
      (buffer[curinput.locfield] = 37) THEN
      GOTO 30;
    IF NOT morename(buffer[curinput.locfield])
      THEN GOTO 30;
    curinput.locfield := curinput.locfield+1;
  END;
30:
  endname;
END;

```

Figure 3: Filename scanning in PASCAL, after manual prettyprinting. The statements `beginname` and `endname` are calls to procedures without arguments. The magic constants 32, 37, and 59 would normally have been given symbolic names, but this code is output by the TANGLE preprocessor which already replaced those names by their numeric values. The lack of statements to exit loops and return from procedures forces programmers to resort to the infamous `goto` statements, which are required to have predeclared numeric labels in PASCAL.

is the biggest single problem in PASCAL. ... I feel that it is a mistake to use PASCAL for anything much beyond its original target. In its pure form, PASCAL is a toy language, suitable for teaching but not for real programming.

There is also a good survey of ambiguities and insecurities of the language by Welsh, Sneeringer, and Hoare [53].

Donald Knuth had co-written a compiler for a subset of ALGOL 60 two decades earlier [2], and had written extensively about that language [41, 21, 20, 22, 24, 25]. Moreover, he had developed the fundamental theory of parsing that is used in compilers [23]. He was therefore acutely aware of the limitations of PASCAL, and to enhance portability of T_EX and METAFONT, and presciently (see Section 8), to facilitate future translation to other languages, sharply restricted his use of features of that language [37, Part 1].

The botch that Brian Kernighan criticized has to do with the fact that in PASCAL, object sizes are part of their type: if you declare a variable to hold ten characters, then it is illegal to assign a string of any other length to it, and if it appears as a routine argument, then all calls to that routine must pass a string of exactly the correct length.

Donald Knuth’s solution to this extremely vexing problem for programs like T_EX and METAFONT that mainly deal with streams of input characters was to not use PASCAL directly, but rather, to delegate the problem of character-string management, and other tasks, to a preprocessor, called TANGLE. This tool, and its companion WEAVE, are fundamental for the notion of *literate programming* that he developed during this work [34, 50].

Because PASCAL had mainly been used for small programs, few compilers for that language were prepared to handle programs as large and complex as T_EX and METAFONT. Their PASCAL source code produced by TANGLE amounts to about 20,000 lines each when prettyprinted.

Ports of T_EX and METAFONT to new systems frequently uncovered compiler bugs or resource limits that had to be fixed before the programs could operate. The 16-bit computers were particularly challenging because of their limited address space, and it was a remarkable achievement when Lance Carnes announced T_EX on the HP3000 in 1981 [5], followed not long after by his port to the IBM PC with the wretched 64KB memory segments of the Intel 8086 processor. He later founded a company, *Personal T_EX, Inc.* About the same time, David Fuchs completed an independent port to the IBM PC, and that effort was briefly available commercially. David Kellerman and Barry Smith left *Oregon Software*, where they worked on PASCAL compilers, to found the company *Kellerman & Smith* to support T_EX in the VAX VMS environment. Barry later started *Blue Sky Research* to support T_EX on the Apple MACINTOSH.

8 Switching languages, again

UNIX users had more of a problem getting T_EX and METAFONT, because of compiler problems. Pavel Curtis and Howard Trickey first announced a port in 1983 [7], where they noted:

Unhappily, the PC compiler has more deficiencies than one might wish.

Their project took several months, and ultimately, they had to make several changes and extensions to the PASCAL compiler.

In 1986–1987, Pat Monardo at the University of California, Berkeley, did the UNIX community a great

service when he undertook a translation, partly machine assisted, and partly manual, of \TeX from PASCAL to C, the result of which he called COMMON \TeX . That work ultimately led to the Web-to-C project to which many people have contributed, and today, virtually all UNIX installations, and indeed, the entire \TeX Live distribution for UNIX and Microsoft WINDOWS, is based on the completely automated translation of the master source files of all \TeX ware and METAFONTware from the Web sources to PASCAL and then to C.

Although we shall not further describe it here, it is worth noting that yet another programming language has since been used to reimplement \TeX : Karel Skoupy's work with JAVA [14].

Another interesting project is Achim Blumensath's *ANT: A Typesetting System* [4], where the recursive acronym means *ANT is not \TeX* . The first version was done in the LISP dialect SCHEME, and the current version is in OCAML. Input is very similar to \TeX markup, and output can be DVI, PostScript, or PDF.

9 Wrapping up

In this article, I have described how architecture, operating systems, programming languages, and resource limits influenced the design of \TeX and METAFONT. This analysis is in no way intended to be critical, but instead, offer a historical retrospective that is, I believe, helpful to think about for other widely-used software packages as well.

\TeX and METAFONT, and the literate programming system in which they are written, are truly remarkable projects in software engineering. Their flexibility, power, reliability, and stability, and their unfettered availability, have allowed them to be widely used and relied upon in academia, industry, and government. Donald Knuth expects to use them for the rest of his career, and so do many others, including this author. His willingness to expose his programs to public scrutiny by publishing them as books [37, 39], and then to further admit to errors in them [33] in order to learn how we might become better programmers, are traits too seldom found in others.

References

- [1] Urs Ammann. On code generation in a PASCAL compiler. *Software—Practice and Experience*, 7(3):391–423, May/June 1977. CODEN SPEXBL. ISSN 0038-0644.
- [2] G. A. Bachelor, J. R. H. Dempster, D. E. Knuth, and J. Speroni. SMALGOL-61. *Communications of the Association for Computing Machinery*, 4(11):499–502, November 1961. CODEN CACMA2. ISSN 0001-0782. URL <http://doi.acm.org/10.1145/366813.366843>.
- [3] J. W. Backus, F. L. Bauer, J. Green, C. Katz, J. McCarthy, A. J. Perlis, H. Rutishauser, K. Samelson, B. Vauquois, J. H. Wegstein, A. van Wijngaarden, and M. Woodger. Revised report on the algorithmic language Algol 60. *Communications of the Association for Computing Machinery*, 6(1):1–17, January 1963. CODEN CACMA2. ISSN 0001-0782. URL <http://doi.acm.org/10.1145/366193.366201>. Edited by Peter Naur. Dedicated to the memory of William Turanski.
- [4] Achim Blumensath. ANT: A typesetting system. World-Wide Web document and software, October 24, 2004. URL <http://www-mgi.informatik.rwth-aachen.de/~blume/Download.html>.
- [5] Lance Carnes. \TeX for the HP3000. *TUGboat*, 2(3):25–26, November 1981. ISSN 0896-3207.
- [6] M. Crispin. RFC 4042: UTF-9 and UTF-18 efficient transformation formats of Unicode, April 2005. URL <ftp://ftp.internic.net/rfc/rfc4042.txt>, <ftp://ftp.math.utah.edu/pub/rfc/rfc4042.txt>. Status: INFORMATIONAL.
- [7] Pavel Curtis and Howard Trickey. Porting \TeX to VAX/UNIX. *TUGboat*, 4(1):18–20, April 1983. ISSN 0896-3207.
- [8] Edward R. Fiala. MAXC systems. *Computer*, 11(5):57–67, May 1978. CODEN CPTRB4. ISSN 0018-9162. URL <http://research.microsoft.com/~lampson/Systems.html#maxc>.
- [9] Michel Goossens, Frank Mittelbach, and Alexander Samarin. *The \LaTeX Companion*. Tools and Techniques for Computer Typesetting. Addison-Wesley, Reading, MA, USA, 1994. ISBN 0-201-54199-8. xxi + 528 pp. LCCN Z253.4.L38 G66 1994.
- [10] Michel Goossens and Sebastian Rahtz. *The \LaTeX Web companion: integrating \TeX , HTML, and XML*. Tools and Techniques for Computer Typesetting. Addison-Wesley Longman, Harlow, Essex CM20 2JE, England, 1999. ISBN 0-201-43311-7. xxii + 522 pp. LCCN QA76.76.H94G66 1999. With Eitan M. Gurari and Ross Moore and Robert S. Sutor.
- [11] Michel Goossens, Sebastian Rahtz, and Frank Mittelbach. *The \LaTeX Graphics Companion: Illustrating Documents with \TeX and PostScript*.

- Tools and Techniques for Computer Typesetting. Addison-Wesley, Reading, MA, USA, 1997. ISBN 0-201-85469-4. xxi + 554 pp. LCCN Z253.4.L38G663 1997.
- [12] Ralph E. Gorin. *Introduction to DECSYSTEM-20 Assembly Language Programming*. Digital Press, 12 Crosby Drive, Bedford, MA 01730, USA, 1981. ISBN 0-932376-12-6. xxx + 545 pp. LCCN QA76.8.D17 .G67.
- [13] Katie Hafner and Matthew Lyon. *Where wizards stay up late: the origins of the Internet*. Simon and Schuster, New York, NY, USA, 1996. ISBN 0-684-81201-0. 304 pp. LCCN TK5105.875.I57H338 1996.
- [14] Hans Hagen. The status quo of the $\mathcal{N}\mathcal{T}\mathcal{S}$ project. *TUGboat*, 22(1/2):58–66, March 2001. ISSN 0896-3207.
- [15] Ken Harrenstien. KLH10 PDP-10 emulator. World-Wide Web document and software, 2001. URL <http://klh10.trailing-edge.com/>. This is a highly-portable simulator that allows running TOPS-20 on most modern Unix workstations.
- [16] John Douglas Hobby. *Digitized Brush Trajectories*. Ph.D. dissertation, Department of Computer Science, Stanford University, Stanford, CA, USA, June 1986. 151 pp. URL <http://wwwlib.umi.com/dissertations/fullcit/8602484>. Also published as report STAN-CS-1070 (1985).
- [17] Brian W. Kernighan. Why Pascal is not my favorite programming language. Computer Science Report 100, AT&T Bell Laboratories, Murray Hill, NJ, USA, July 1981. URL <http://cm.bell-labs.com/cm/cs/cstr/100.ps.gz>. Published in [18].
- [18] Brian W. Kernighan. Why Pascal is not my favorite programming language. In Alan R. Feuer and Narain Gehani, editors, *Comparing and assessing programming languages: Ada, C, and Pascal*, Prentice-Hall software series, pages 170–186. Prentice-Hall, Englewood Cliffs, NJ, USA, 1984. ISBN 0-13-154840-9 (paperback), 0-13-154857-3 (hard). LCCN QA76.73.A35 C66 1984. See also [17].
- [19] Brian W. Kernighan and P. J. Plauger. *Software Tools in Pascal*. Addison-Wesley, Reading, MA, USA, 1981. ISBN 0-201-10342-7. ix + 366 pp. LCCN QA76.6 .K493.
- [20] D. E. Knuth, L. L. Bumgarner, D. E. Hamilton, P. Z. Ingerman, M. P. Lietzke, J. N. Merner, and D. T. Ross. A proposal for input-output conventions in ALGOL 60. *Communications of the Association for Computing Machinery*, 7(5):273–283, May 1964. CODEN CACMA2. ISSN 0001-0782. URL <http://doi.acm.org/10.1145/364099.364222>. Russian translation by M. I. Ageev in *Sovremennoe Programirovanie* 1 (Moscow: Soviet Radio, 1966), 73–107.
- [21] Donald E. Knuth. Man or boy? *Algol Bulletin (Amsterdam: Mathematisch Centrum)*, 17(?):7, January 1964. CODEN ALGOBG. ISSN 0084-6198.
- [22] Donald E. Knuth. Man or boy? *Algol Bulletin (Amsterdam: Mathematisch Centrum)*, 19(7):8–9, January 1965. CODEN ALGOBG. ISSN 0084-6198.
- [23] Donald E. Knuth. On the translation of languages from left to right. *Information and Control*, 8(6):607–639, December 1965. CODEN IFCNA4. ISSN 0019-9958. Russian translation by A. A. Muchnik in *Īazyki i Avtomaty*, ed. by A. N. Maslov and Ē. D. Stotskiĭ (Moscow: Mir, 1975), 9–42. Reprinted in *Great Papers in Computer Science* (1996) [?].
- [24] Donald E. Knuth. Teaching ALGOL 60. *Algol Bulletin (Amsterdam: Mathematisch Centrum)*, 19(?):4–6, January 1965. CODEN ALGOBG. ISSN 0084-6198.
- [25] Donald E. Knuth. The remaining trouble spots in ALGOL 60. *Communications of the Association for Computing Machinery*, 10(10):611–618, October 1967. CODEN CACMA2. ISSN 0001-0782. URL <http://doi.acm.org/10.1145/363717.363743>. Reprinted in E. Horowitz, *Programming Languages: A Grand Tour* (Computer Science Press, 1982), 61–68.
- [26] Donald E. Knuth. *Fundamental Algorithms*, volume 1 of *The Art of Computer Programming*. Addison-Wesley, Reading, MA, USA, 1968. ISBN 0-201-03803-X. xxi + 634 pp. LCCN QA76.5 .K74. Second printing, revised, July 1969, with page count xxi + 634.
- [27] Donald E. Knuth. *Seminumerical Algorithms*, volume 2 of *The Art of Computer Programming*. Addison-Wesley, Reading, MA, USA, 1969. ISBN 0-201-03802-1. xi + 624 pp. LCCN QA76.5 .K57.
- [28] Donald E. Knuth. *Seminumerical Algorithms*, volume 2 of *The Art of Computer Programming*. Addison-Wesley, Reading, MA, USA, 1971. ISBN 0-201-03802-1. xii + 624 pp. LCCN QA76.5 .K57. Second printing, revised, November 1971.

- [29] Donald E. Knuth. *Fundamental Algorithms*, volume 1 of *The Art of Computer Programming*. Addison-Wesley, Reading, MA, USA, second edition, 1973. ISBN 0-201-03809-9. xxi + 634 pp. LCCN QA76.6 .K641 1973. Second printing, revised, February 1975.
- [30] Donald E. Knuth. *Sorting and Searching*, volume 3 of *The Art of Computer Programming*. Addison-Wesley, Reading, MA, USA, 1973. ISBN 0-201-03803-X. xii + 722 pp. LCCN QA76.5 .K74.
- [31] Donald E. Knuth. *Sorting and Searching*, volume 3 of *The Art of Computer Programming*. Addison-Wesley, Reading, MA, USA, March 1975. ISBN 0-201-03803-X. xii + 725 pp. LCCN QA76.5 .K74. Second printing, revised.
- [32] Donald E. Knuth. *\TeX and METAFONT—New Directions in Typesetting*. Digital Press, 12 Crosby Drive, Bedford, MA 01730, USA, 1979. ISBN 0-932376-02-9. xi + 201 + 105 pp. LCCN Z253.3 .K58 1979.
- [33] Donald E. Knuth. The errors of \TeX . Technical Report STAN-CS-88-1223, Stanford University, Department of Computer Science, September 1988. See [33].
- [34] Donald E. Knuth. The errors of \TeX . *Software—Practice and Experience*, 19(7):607–685, July 1989. CODEN SPEXBL. ISSN 0038-0644. This is an updated version of [?]. Reprinted with additions and corrections in [34, pp. 243–339].
- [35] Donald E. Knuth. *Literate Programming*. CSLI Lecture Notes Number 27. Stanford University Center for the Study of Language and Information, Stanford, CA, USA, 1992. ISBN 0-937073-80-6 (paper), 0-937073-81-4 (cloth). xv + 368 pp. LCCN QA76.6.K644.
- [36] Donald E. Knuth. *Digital Typography*. CSLI Publications, Stanford, CA, USA, 1999. ISBN 1-57586-011-2 (cloth), 1-57586-010-4 (paperback). xvi + 685 pp. LCCN Z249.3.K59 1998.
- [37] Donald E. Knuth. *The \TeX book*, volume A of *Computers and Typesetting*. Addison-Wesley, Reading, MA, USA, 1986. ISBN 0-201-13447-0. ix + 483 pp. LCCN Z253.4.T47 K58 1986.
- [38] Donald E. Knuth. *\TeX : The Program*, volume B of *Computers and Typesetting*. Addison-Wesley, Reading, MA, USA, 1986. ISBN 0-201-13437-3. xv + 594 pp. LCCN Z253.4.T47 K578 1986.
- [39] Donald E. Knuth. *The METAFONTbook*, volume C of *Computers and Typesetting*. Addison-Wesley, Reading, MA, USA, 1986. ISBN 0-201-13445-4. xi + 361 pp. LCCN Z250.8.M46 K58 1986.
- [40] Donald E. Knuth. *METAFONT: The Program*, volume D of *Computers and Typesetting*. Addison-Wesley, Reading, MA, USA, 1986. ISBN 0-201-13438-1. xv + 560 pp. LCCN Z250.8.M46 K578 1986.
- [41] Donald E. Knuth. *Computer Modern Typefaces*, volume E of *Computers and Typesetting*. Addison-Wesley, Reading, MA, USA, 1986. ISBN 0-201-13446-2. xv + 588 pp. LCCN Z250.8.M46 K574 1986.
- [42] Donald E. Knuth and Jack N. Merner. ALGOL 60 confidential. *Communications of the Association for Computing Machinery*, 4(6):268–272, June 1961. CODEN CACMA2. ISSN 0001-0782. URL <http://doi.acm.org/10.1145/366573.366599>.
- [43] Leslie Lamport. *\LaTeX —A Document Preparation System—User’s Guide and Reference Manual*. Addison-Wesley, Reading, MA, USA, 1985. ISBN 0-201-15790-X. xiv + 242 pp. LCCN Z253.4.L38 L35 1986.
- [44] Leslie Lamport. *\LaTeX : A Document Preparation System: User’s Guide and Reference Manual*. Addison-Wesley, Reading, MA, USA, second edition, 1994. ISBN 0-201-52983-1. xvi + 272 pp. LCCN Z253.4.L38L35 1994.
- [45] Phillip Laplante, editor. *Great papers in computer science*. IEEE Computer Society Press, 1109 Spring Street, Suite 300, Silver Spring, MD 20910, USA, 1996. ISBN 0-314-06365-X (paperback), 0-07-031112-4 (hardcover). iv + 717 pp. LCCN QA76 .G686 1996. URL <http://bit.csc.lsu.edu/~chen/GreatPapers.html>.
- [46] Franklin Mark Liang. Word hy-phen-a-tion by com-pu-ter. Technical Report STAN-CS-83-977, Stanford University, Stanford, CA, USA, August 1983.
- [47] Frank Mittelbach, Michel Goossens, Johannes Braams, David Carlisle, Chris Rowley, Christine Detig, and Joachim Schrod. *The \LaTeX Companion*. Tools and Techniques for Computer Typesetting. Addison-Wesley, Reading, MA, USA, second edition, 2004. ISBN 0-201-36299-6. xxvii + 1090 pp. LCCN Z253.4.L38 G66 2004.
- [48] Sao Khai Mong. A Fortran version of METAFONT. *TUGboat*, 3(2):25, October 1982. ISSN 0896-3207.

- [49] William M. Newman and Robert F. Sproull. *Principles of Interactive Computer Graphics*. McGraw-Hill Computer Science Series, Editors: Richard W. Hamming and Edward A. Feigenbaum. McGraw-Hill, New York, NY, USA, 1973. ISBN 0-07-046337-9. xxviii + 607 pp. LCCN T385 .N48.
- [50] Michael F. Plass. *Optimal pagination techniques for automatic typesetting systems*. Thesis (ph. d.), Stanford University, Stanford, CA, USA, 1981. vi + 72 pp.
- [51] Lynn Elizabeth Ruggles. *Paragon, an interactive, extensible, environment for typeface design*. Ph.D. dissertation, University of Massachusetts Amherst, Amherst, MA, USA, 1987. 192 pp. URL <http://wwwlib.umi.com/dissertations/fullcit/8805968>.
- [52] E. Wayne Sewell. *Weaving a Program: Literate Programming in WEB*. Van Nostrand Reinhold, New York, NY, USA, 1989. ISBN 0-442-31946-0. xx + 556 pp. LCCN QA76.73.W24 S491 1989.
- [53] Guy L. Steele Jr. *Common Lisp—The Language*. Digital Press, 12 Crosby Drive, Bedford, MA 01730, USA, 1984. ISBN 0-932376-41-X. xii + 465 pp. LCCN QA76.73.L23 S73 1984. US\$22.00.
- [54] Guy L. Steele Jr. *Common Lisp—The Language*. Digital Press, 12 Crosby Drive, Bedford, MA 01730, USA, second edition, 1990. ISBN 1-55558-041-6 (paperback), 1-55558-042-4 (hardcover), 0-13-152414-3 (Prentice-Hall). xxiii + 1029 pp. LCCN QA76.73.L23 S73 1990. See also [51].
- [55] J. Welsh, W. J. Sneeringer, and C. A. R. Hoare. Ambiguities and insecurities in Pascal. *Software—Practice and Experience*, 7(6):685–696, November/December 1977. CODEN SPEXBL. ISSN 0038-0644.
- [56] Niklaus Wirth. The design of a PASCAL compiler. *Software—Practice and Experience*, 1(4): 309–333, October/December 1971. CODEN SPEXBL. ISSN 0038-0644.
- [57] Niklaus Wirth. *Algorithms + Data Structures = Programs*. Prentice-Hall Series in Automatic Computation. Prentice-Hall, Englewood Cliffs, NJ, USA, 1976. ISBN 0-13-022418-9. xvii + 366 pp. LCCN QA76.6 .W561.
- [58] F. H. G. Wright II and R. E. Gorin. *FAIL*. Computer Science Department, Stanford University, Stanford, CA, USA, May 1974. Stanford Artificial Intelligence Laboratory Memo AIM-226 and Computer Science Department Report STAN-CS-74-407.
- [59] Ignacio Andres Zaballa Salelles. *Interfacing with graphics objects*. PhD thesis, Department of Computer Science, Stanford University, Stanford, CA, USA, December 1982. 146 pp.

A L^AT_EX fledgling struggles to take flight

Peter L. Flom

National Development and Research Institutes, Inc.
71 West 23rd St., 8th floor
New York, NY 10010*

A little about this article

I work as a statistical consultant and data analyst at a nonprofit research company. I also work as an independent statistical consultant, mostly to graduate students in the social and behavioral sciences. I've done almost no computer programming (I did have one course in computer programming, but it was so long ago that we used punch cards and waited a day or more for our programs to run on the mainframe that took up most of the basement; I also write some very simple programs in R).

When I read the first issue of *The PracT_EX Journal*, I was thrilled. Finally, someone was writing a journal for beginners. So, I wrote a very enthusiastic 'Thank you' to the editor (Lance Carnes), and he wrote back, thanking me for the feedback, and asking me to write an article. I said OK. And here it is.

I'm writing with two groups in mind: Beginners, and people who write for beginners. I'd like to offer both groups some perspective from someone who is just a little way along the path. I'd like to let the true beginners know that it is possible to learn L^AT_EX; after only a few months of intermittent use, I can do a lot — I have written entire articles in L^AT_EX, some of them with quite complicated organizational structure and with fairly intimidating formulas; I've also started doing some presentations in L^AT_EX, using the Beamer package. If I can do it, you can. I'd like to give the teachers the perspective of a recent beginner, so that their efforts can have maximum reward; when I consider that so many people contribute to L^AT_EX, often without any monetary reward, I imagine that those people would like to have their efforts help as many people as possible to use L^AT_EX easily and well. This article is in three sections:

1. Introduction
2. Some suggestions for teaching beginners
3. Some hints for beginners

* Work on this project was supported by NIDA grant P30 DA11041; I'd like to thank the editor and the reviewers for their helpful comments and encouragement. The article was originally published in *The PracT_EX Journal*, issue 2005-2, <http://tug.org/pracjourn>.

I hope, however, that both teachers and learners will read all the sections — the division of material is not rigid.

How I started using L^AT_EX

Long ago, I used Nota Bene. This was a very nice word processing program, designed for use by scholars. But no one I knew used it, so . . . I then became a dissatisfied user of Microsoft Word for years. But it came with my computer, everyone else used it, journal editors liked it, so, I used it. Then, at the recommendation of a friend and colleague, I started using WinEdt to write R files (R is a language and environment for statistical computing and graphics). It's great for that purpose, but I noticed that it kept mentioning L^AT_EX. I looked into it a little, but it looked really hard, so I didn't do much.

Then, I saw on the R-help list that someone was writing a book on R for beginners. I asked if he wanted some help from a beginner. He said he did; but the files were in L^AT_EX. He expressed amazement that I didn't use it. But it looked really hard, so I didn't do much.

Then I wrote a grant proposal that included a lot of formulas. A consultant on the grant did not have Word on his machine. He recommended L^AT_EX; but my co-investigator wanted Word files. So, I started looking more into L^AT_EX, and into programs to convert Word into L^AT_EX and vice versa. The deadline was looming, so I wrote that grant in Word (using Math Edit), and wrote files out as rtf files, which my consultant could read. Still, some formulas didn't print right; or printed differently on different computers; it was a mess. So, I resolved to learn L^AT_EX. I've been using it more and more over the last 6 months or so, and now really prefer it to Word, for virtually everything. Maybe after reading another issue of this journal, I'll prefer it for absolutely everything.

Some suggestions for teaching beginners

Ease of use

L^AT_EX looks hard. When I first saw a .tex file, I wondered how anyone could ever learn to write such

stuff. There are reasons for this: \LaTeX was (naturally) written and extended by computer scientists (Donald Knuth for \TeX , Leslie Lamport for \LaTeX , and many others), and that's probably why it looks like a programming language.² When you are really expert at something, it's hard to remember what it was like to not be expert; when you are really talented at something, it's hard to empathize with the less talented — this is not to criticize the people who write for beginners, it's just the way people are. Well, I am neither experienced nor talented at programming, so I *can* empathize; even moderately complex \LaTeX files look indecipherable to true beginners (at least, they did to me). Part of this is due to how people are first exposed to \LaTeX . The first `.tex` file I saw was one which was going to be a book on a statistical programming language. I think that many people who start using \LaTeX do so because of the limitations of Word or Word Perfect, or some other program. Thus, the first things we want to write are complicated files. Also, for the people who write documentation, it's easy to get into tricky stuff quickly, and this makes sense — there's not much point in having pages and pages of very simple documents.

One way of making the learning curve a little less steep is to provide annotated programs. Another might be to provide more exercises and treat an introductory book more as a text.

So, if you're writing for true beginners, emphasize ease of use. And, as \LaTeX becomes used by more people who are not and never were programmers, try to remember that we don't think the way you think. If you're a programmer who doesn't like statistics, maybe thinking about how you would like to learn statistics would help in how people like me like to learn things like \LaTeX .

Distributions

Everything I see on \LaTeX mentions several (or more than several) different distributions. This just confused the heck out of me. Is there a difference? (I still don't know). Is one better than the other? (I still don't know). Some are free, some are commercial — what advantages do the commercial programs have? (They must have *some* or the companies would go out of business). I've heard about LyX, which is a WYSIWYG version of \LaTeX — this seems nice, but what are the drawbacks? I wound up using \TeX Live, more or less by chance. Now I use `proTeXt`, because that's what I got sent as a

² Reviewers pointed out that most all document markup languages developed in the pre-GUI (graphical user interface) era looked like this.

member of the \TeX Users Group (TUG). It would be good if some documentation could list the various distributions and what their strengths are, or state that there are no real differences.

Writing in \LaTeX is not like writing in Word

In Word (and probably in other word processors) when you don't get what you want, it's often because the program is illogical. It does some things automatically, some (most?) of which make no sense. In \LaTeX , though, when you don't get what you want it's often because you messed up. When I started writing things that were a little complex, I often got errors. This still happens. At first, this really annoyed me. It almost made me stop using \LaTeX . But I realized I should look on this more like a programming problem: Debugging is often necessary, and this doesn't mean you're stupid. I got this from the minimal programming I've done in R, but others who have never done any programming at all may not get this attitude, and I didn't see it in any of what I've read. Programmers may be so used to this way of thinking that they don't even mention it.

Adding packages

I find this very confusing.³ I've read various help files on how to do this; I'm sure they're all correct, I know they're all written by experts. It seems to me, as a nonprogrammer, that they contradict each other. I know they really don't, because then they wouldn't all work. So, it must be that I am even more confused than I thought, which is saying something. I don't fully understand *why* this has to be so hard (like I said, I am no programmer). The other free software I use a lot is R, which also runs on lots of platforms, and also has lots of additional packages written by lots of different people, but there, when you add a package, it does all the background work for you, you just find the package you want, click on it, and you're done. If it can't be made easy, then I would strongly urge recommending that beginners install everything — all the available packages — at once. Disk space is cheap, writing the files takes a while, but it only needs to be done once. That's what I wound up doing (by uninstalling all the files, and then reinstalling everything I could get all at once) and this worked perfectly.

To a large extent, these problems have been solved by `proTeXt`, which automates a lot of this. But, as far as I know, it is only for Windows, and

³ According to one reviewer, this may not be as difficult as I think it is — there are, apparently, tools for doing this that I am unaware of; I am just writing about what I know.

L^AT_EX users using other systems may still have the type of question outlined above.

Annotated programs

All the books and other material on learning L^AT_EX include numerous examples of L^AT_EX files, which is good. One of the best ways of learning is by example. But one way to make these examples even more useful would be to include extensive annotations, either in the margins, in footnotes, or in text immediately below the program. What I have in mind is something like the way many editions of Shakespeare have notes explaining terms and references that are unfamiliar. The first few times a command is used, it would be useful to include a note. Kopka and Daly (3) do a nice job of this in their “Sample L^AT_EX file” on pages 16–19; I’d like to see more examples like this.

Debugging and error messages

Whenever I do anything complicated in L^AT_EX (and sometimes when I do something simple) I get errors. The messages accompanying these are sometimes helpful, but often rather obscure, at least to non-programmers such as myself. It would be great to have a source that explains some of these error messages in ordinary English. It would also be great to have some reference on debugging.⁴

Some hints for beginners

L^AT_EX has to be *learned*

Word is designed not to be learned. It’s supposed to function right out of the box (whether it does or not is another matter); if you are used to Word, then you may think that you should be able to use L^AT_EX right out of the box. Well, maybe some people can. I couldn’t. On the other hand, as you learn L^AT_EX, you get more and more control over how your document looks.

Some resources

There are a *lot* of free resources available for L^AT_EX (see the CTAN website). A lot of these are wonderful, and some are intended for beginners. I know some people find these resources to be enough for them to use L^AT_EX very well. Personally, I like books. I keep three close at hand: *Math into L^AT_EX* (1) is on my desk, and *Guide to L^AT_EX* (3), and *The L^AT_EX Companion* (4) are on my bookshelf. I like books (as opposed to web-based material) in general because

⁴ I have since found that Kopka and Daly (3) do include a list of some error messages in an appendix.

1. They have extensive tables of contents and indexes
2. They are already bound and thus easy to flip through
3. I am just old-fashioned enough to like being able to page through a book, and keep it open on my desk while I work on something complex.

I like the three books mentioned above for different things. The L^AT_EX Companion (4) is a great book, but not for beginners. It’s intimidating. It’s too big. It assumes knowledge. I think it should be the 3rd or 4th book a L^AT_EX user buys; it’s a great reference, but it still kind of intimidates me. Kopka and Daly’s *Guide to L^AT_EX* (3) is the best introduction to L^AT_EX that I’ve seen. The book I use most now is George Grätzer’s *Math into L^AT_EX* (1) (it’s open on my desk as I write this, I just looked up how to type the author’s accented name). I use this all the time, partly because one of the main reasons I started using L^AT_EX was to typeset some complex mathematical formulas. All three of these books are very well organized and comprehensible, given their depth. Your taste in particular books may vary. Try out a few. Even if you buy a bunch of books before finding one or two you really like, it’s not that much money (after all, the software is free).

Another resource I find very helpful is the T_EX users mailing list; information on this group is at <http://tug.org/mailman/listinfo/texhax>.

Figure out what you need to know, and when you need to know it

L^AT_EX is huge. It does all kinds of things, plus a lot that I am sure I am unaware of. What you need from it depends on what kind of work you do. For instance, I need to do a lot with tables, equations, bibliographies and imported graphics; I had to learn these first. But I don’t have as much need to make my own drawings — I’ll wait. Learning about some different fonts would be fun, but not urgent (for *me* — this may be very urgent for *you*). I will probably never learn to typeset Sanskrit or musical notation. But just figuring out what is available can be a challenge. One thing to do, after you can write basic documents, is to browse through various sources, including books and the CTAN website; Jim Hefferon wrote a good introduction to the website in the first issue of this journal (2). Try to follow the discussions on the mailing group. This journal, of course, is very helpful; and then there’s *TUGboat*, which is mostly more advanced (sometimes, I don’t even understand the titles!).

Run files often

Run your file through L^AT_EX a lot. Each time you do something even a little interesting, where you have any doubts about whether what you are doing will work correctly, typeset the file. If you've only made one or a few changes since you last ran the file, then it will be easier to find your error. In the editor I use (WinEdt) you can also typeset a small part of your document (hit ctrl+shift+c). This saves a lot of time. On a related note, make backups often, and give them names you will understand and remember later. In particular, if you've gotten something complicated to work reasonably well, but still want to tweak it a little, save the file that works before you forget how you got it to work. (For me, this happens most often with complex, multiline equations and with tables that have complicated structures).

Look at examples

All the books I listed have *lots* of examples. Try to figure out how they work and how they could be changed. Fool around; see what happens.

Make a default preamble

As you learn more L^AT_EX, you will (probably) find that there are certain packages that you always want loaded. It's hard (at least for me) to remember which ones I want, so I made a default file;⁵ as of March 6, 2005, it looked like this:

```
\documentclass{article}
\usepackage{graphicx}
\usepackage{amsmath, amssymb, latexsym, amsthm}
\usepackage{exscale, mathrsfs}
\usepackage{caption2, float, chapterbib, natbib}
\usepackage[section]{placeins}
\usepackage{fancyhdr}
\usepackage{geometry}
\usepackage[symbol, perpage]{footmisc}

\theoremstyle{plain}
\newtheorem{theorem}{Theorem}

\theoremstyle{definition}
\newtheorem{definition}{Definition}

\begin{document}
\title{Put title here}
\author{Peter L. Flom}
\maketitle
Sample text
\bibliographystyle{amsplain}
\bibliography{file name}
\end{document}
```

⁵ One of the reviewers commented that it would be better to make a .sty file; I, however, do not know how to do this.

Summary

As I get more and more used to L^AT_EX, I find it more and more useful. I am gradually using it for more and more documents. For me, the best things about using L^AT_EX, as opposed to Word, are

1. L^AT_EX directs my attention to things that need attention. It takes care of section formatting, typography, and so on; but it forces my attention to things like complicated mathematical formulas and complex tables.
2. The ability to typeset complex mathematical equations and know they will appear correctly on other people's computers and on printout.
3. The naturalness of section formatting (such as with `\section`, and related commands)
4. The ease of cross-referencing to different sections of a document (using `\label` and `\ref`).
5. The helpfulness of the L^AT_EX community in finding solutions.

The biggest barriers to using L^AT_EX are

1. Working with co-authors and editors who insist on Word files.
2. Formatting complex tables.
3. Learning to use my editor (WinEdt) more efficiently.
4. Remembering that getting an error message is not the computer telling me that I am stupid (careless, ignorant, forgetful . . . but not stupid).

I look forward to learning more, and to becoming more expert, and to finding ways to spread my L^AT_EX wings. Certainly writing this article helped me do so, I hope reading it helped you, as well.

Bibliography

- [1] George Grätzer, *Math into L^AT_EX*, Birkhäuser, New York, 2000.
- [2] Jim Hefferon, *CTAN for starters*, The PracT_EX Journal **1** (2005).
- [3] Helmut Kopka and Patrick W. Daly, *Guide to L^AT_EX*, Addison Wesley, Boston, 2004.
- [4] Frank Mittelbach and Michel Goossens, *The L^AT_EX companion*, Addison Wesley, Boston, 2004.

Strategies for including graphics in \LaTeX documents

Klaus H\"oppner

Nieder-Ramstädter Str. 47

64283 Darmstadt

Germany

klaus.hoepfner@gmx.de

Abstract

This talk presents strategies for including graphics into \LaTeX documents. It shows the usage of the standard `graphics` packages of \LaTeX as well as an introduction to different graphics formats. Some external tools for converting graphics formats are shown.

Overview of graphics formats

In general, there exist two kinds of graphics formats: vector and bitmap graphics. For bitmapped images there exist different flavors, no compression (what makes your files really big dependent on resolution and color depth, so I won't cover them from here on), compression methods that just do data compression and preserve the images's quality and lossy compression methods with a reduction of images's quality.

So let's go more into detail:

Vector graphics are set up by drawing or filling geometrical objects like lines, Bezi\er curves, polygons, circles and so on. The properties of these objects are stored mathematically. Vector graphics are in general device independent. It is easy to scale or rotate them without loss of quality since the job to rasterize them is done by the printer or printer driver.

Bitmaps without lossy compression store the image information as pixels where each pixel is given a color. In principle the quality of a bitmap becomes better the higher the resolution and color depth is (e.g. GIF files use a color

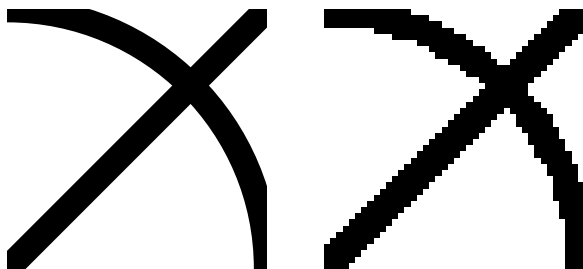


Figure 1: Zoomed view into a sample image as vector graphics (left) and bitmap (right).

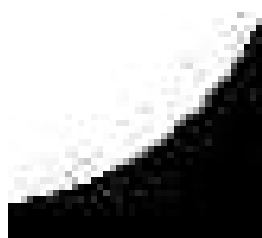


Figure 2: A low quality JPEG image showing some artifacts at the transition between black and white

depth of 8 bit leading to 256 different indexed colors while a bitmap with 24 bit color depth can have about 16 millions colors). Scaling and rotating bitmap images will yield in a loss of quality, and printing bitmaps to a device with a different resolution can produce bad results. Fig. 1 shows the difference between a scaled image as vector and bitmap graphics.

Bitmaps with lossy compression uses the fact that the human eye is fairly good at seeing small differences in brightness over a relatively large area, but not so good at distinguishing the exact strength of a high frequency brightness variation. For this reason, components in the high frequency region are reduced leading to smaller file sizes. This works well for photographs that usually contain smooth transitions in color, but for graphics with sharp border artifacts can occur as shown in fig. 2. The most prominent graphics format using lossy compression are JPEG files.

Graphics formats in practice

There exist many graphics formats, so I will concentrate on some of them that are often used:

EPS is the encapsulated PostScript format. It is used for vector graphics but can also contain bitmap graphics.

PNG is the portable network graphics format. It was introduced due to the problem that Unisys claimed a patent for the compression algorithm used in GIF. For this reason, it is often used nowadays on WWW pages. PNG is a bitmap format that supports both compression without and with quality loss of the image.

JPEG is a bitmap format with lossy compression and is often used for photographs (e.g. most digital cameras produce JPEG files).

TIFF is a bitmap format often used for high quality pictures — not only because it supports the CMYK color space for printing.

Now the question is: What format shall I use for what purpose? Though there is no definite answer to this question, my advice is as follows:

1. For drawings (e.g. technical drawings or data plots) use vector graphics. It gives you maximum freedom on manipulating the the image when including it into a document where you may want to scale the image to fit into your layout. Additionally, you are independent of your output device, and you can zoom into the image in your document viewer without seeing single pixels. The drawing tools offered by TeX distributions — PSTricks and METAPOST — produce EPS output natively. Most vector drawing programs like xfig and Corel Draw offer export functionality for producing EPS output (though sometimes buggy).
2. If you are stuck to bitmaps, use PNG for images with sharp color transitions.
3. For photographs, you can use JPEG in most cases, since the quality loss by compression normally doesn't matter in printout. On most devices, a resolution of 100 to 200 dpi will be sufficient (remember that screen resolution is normally about 75 to 100 dpi, and color printers claim to have high resolutions but dither color prints, so you will hardly notice the difference compared to JPEGs with higher resolution).

The L^AT_EX graphics package

Since the introduction of L^AT_EX 2_ε, the `graphics` bundle is part of the standard packages accompanying the L^AT_EX base distribution [1]. It consists of two style files, `graphics.sty` and `graphicx.sty`. While `graphics.sty` requires the use of `\scalebox` and `\rotatebox` for scaling or rotating your graphics, the extended style `graphicx.sty` supports scaling and rotating using the `keyval` package. In general, there is no reason not to always use `graphicx.sty`.

So the first step is to load the `graphicx` style file after the `\documentclass` statement:

```
\usepackage{graphicx}
```

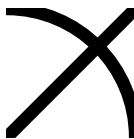
The `graphics` bundle is driver dependent, i.e. it needs to know the output driver it has to produce `\special` statements for. Nowadays, there are two main workflows for producing documents: using `latex` to produce a DVI file and then `dvips` for converting it to PostScript, and using `pdflatex` to produce a PDF file. Most modern T_EX systems are configured to automatically check whether you are using `latex` or `pdflatex` and creating `dvips \specials` in the first case and producing the appropriate `\pdfimage` statements in the second case. So if you are using one of the above workflows, you shouldn't need to specify your output backend explicitly. If you are using another backend you have to specify it as an option, e.g.

```
\usepackage[dvipsone]{graphicx}
```

but be aware that other backends often don't support scaling or rotating.

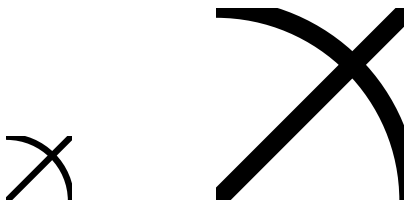
To include an image simply use

```
\includegraphics{sample1}
```



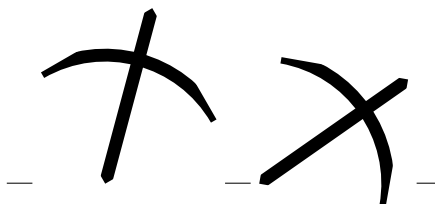
Please notice that no extension for the file was given. The explanation why will follow later. In the case of using `\includegraphics` without options the image is included in its natural size. When using the `graphicx` style, you can scale your image by a factor:

```
\includegraphics[scale=0.5]{sample1}
\includegraphics[scale=1.5]{sample1}
```



Another option supports rotating an image:

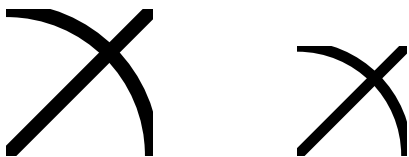
```
\includegraphics[angle=30]{sample1}
\includegraphics[angle=-10]{sample1}
```



Positive numbers lead to counterclockwise rotation, negative numbers to clockwise rotation. The origin for the rotation is the lower left corner of the image, so in counterclockwise rotation in the former example the result does not only have a height but also a depth below the baseline (as shown by the rules).

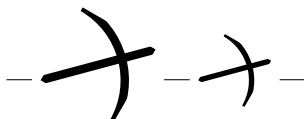
Images can not only be scaled by a given factor, you can specify a height and/or width for the resulting image instead:

```
\includegraphics[width=2cm]
  {sample1}
\includegraphics[height=1.5cm]
  {sample1}
```



`height` gives the height above the baseline. If your image has a depth, you can use `totalheight` instead, i.e. the sum of height and depth will be scaled to the given length.

```
\includegraphics[angle=-30,height=1cm]
  {sample1}
\includegraphics[angle=-30,
  totalheight=1cm]{sample1}
```



You can both specify `width` and `height`. In this case your image may be scaled different in horizontal and vertical direction, unless you use the `keepaspectratio` option:

```
\includegraphics[width=2.5cm,height=1.5cm]
  {sample1}
\includegraphics[width=2.5cm,height=1.5cm,
  keepaspectratio]{sample1}
```



Please notice that using `angle` and `width` or `height` are sensitive for the order you are using the options. Specifying the angle first means that your image is rotated first and then the rotated image is scaled to the desired width or height, while specifying a width or height first will first scale the unrotated image and rotate it afterwards.

Source	Target	Tool
latex+dvips		
EPS		directly supported
PNG	EPS	ImageMagick/netpbm
JPEG	EPS	ImageMagick/netpbm
TIFF	EPS	ImageMagick/netpbm/tif2eps
pdflatex		
PDF		directly supported
EPS	PDF	epstopdf
PNG		directly supported
JPEG		directly supported
TIFF	PNG	ImageMagick/netpbm
TIFF	PDF	tif2eps+epstopdf

Table 1: Conversion to graphics formats supported by `latex` with `dvips` and `pdflatex`

Supported graphics formats

To make things a bit more complicated, `latex` with `dvips` and `pdflatex` support different graphics formats:

`latex+dvips` EPS

`pdflatex` PDF, PNG, JPEG

Table 1 shows ways to convert the standard graphics formats to supported formats.

Fortunately, converting the graphics from existing documents with included EPS graphics from the `latex+dvips` to `pdflatex` workflow is quite easy using the `epstopdf` Perl script that uses Ghostscript to convert EPS to PDF.

It is in general recommended to give the file names in the `\includegraphics` command without extensions. In this case the `graphics` package looks for a supported graphics format automatically. So if you have an image both as EPS and e.g. PDF, you can use both the `latex+dvips` and `pdflatex` workflows without changing the your document.

By the way, including the output of METAPOST is very easy since though it is a variant of EPS, `pdflatex` supports the inclusion of METAPOST output directly. The only thing you have to do is to change the file extension of the output file to `.mps`.

Tools for image conversion

There exist several tools for conversion of graphics formats, both free and commercially. Besides free GUI based tools like Gimp on Unix systems there are two command line tools available for Unix and Windows: ImageMagick [2] and netpbm [3].

ImageMagick can convert images directly, e.g. by typing

```
convert sample.gif sample.png
while netpbm uses the pnm format as intermediate
format:
giftopnm sample.gif | pnmtopng -> sample.png
```

A nice tool is `tif2eps` by Bogusław Jackowski et al. [4] that uses Ghostscript to convert a TIFF file to EPS, e.g.

```
gs -- tif2eps.ps sample.tif sample.esp -rh
```

what produces a RLE compressed and hex encoded EPS file. Upon my experience EPS files produced with `tif2eps` are smaller than those produced by `ImageMagick`. Additionally it supports CMYK TIFF files smoothly.

Additional tools

There are some other tools that are helpful in some cases. I will present two I use quite often.

`overpic` is a \LaTeX package written by Rolf Niepraschk [5]. It includes an image into a \LaTeX picture environment, giving you the opportunity to add new elements into the image with normal \LaTeX picture commands. Fig. 3 shows a map overlaid with symbols and text at some points. The source code for this picture looks like

```
\usepackage[abs]{overpic}
...
\begin{document}
\begin{overpic}[grid,tics=5]{map}
```

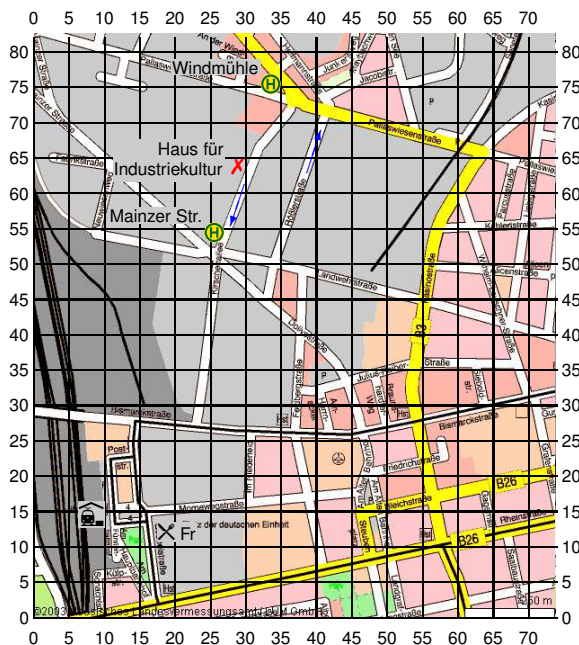


Figure 3: A map with additional marks produced with `overpic`



Figure 4: A bitmap (left) converted to vector graphics (right)

```
\put(32,74){\includegraphics[scale=.3]
{busstop.mps}}
\put(32,77){\llap{\scriptsize%
\colorbox{back}{Windm\`uhle}}}
\put(28,63){\small\textcolor{red}{%
\ding{55}}}
...
\put(17.5,11){\scriptsize\colorbox{back}%
{\Pisymbol{ftsy}{65} Fr}}
\put(6.3,13){\colorbox{back}%
{\Pisymbol{ftsy}{68}}}}
\put(29.8,61.4){\color{blue}\vector(-1,-3){2}}
\put(38.6,63){\color{blue}\vector(1,3){2}}
\end{overpic}
\end{document}
```

`potrace` is a tool to convert a pure black and white bitmap to vectorgraphics [6]. Fig. 4 shows a sample bitmap converted to a vector image.

References

- [1] [CTAN:/macros/latex/required/graphics](http://ctan.org/macros/latex/required/graphics)
- [2] <http://www.imagemagick.org>
- [3] <http://netpbm.sourceforge.net>
- [4] [CTAN:/support/pstools/tif2eps](http://ctan.org/support/pstools/tif2eps)
- [5] [CTAN:/macros/latex/contrib/overpic](http://ctan.org/macros/latex/contrib/overpic)
- [6] <http://potrace.sourceforge.net>

Word to L^AT_EX for a Large, Multi-Author Scientific Paper

D. W. Ignat
PO Box 1380
Middlebury, VT 05753 USA
ignat at mailaps dot org

Abstract

Multiple authors from diverse locations submitted to a scientific journal a manuscript for large review article in many sections, each formatted in MS Word. Journal policy for reviews, which attract no page charges, required a translation to L^AT_EX, including the transformation of section-based references to a non-repetitive article-based list. Saving Word files in RTF format and using `rtf2latex2e` accomplished the basic translation, and then a `perl` program was used to get the references into acceptable condition. This approach to conversion succeeded and may be useful to others.

Introduction

Twelve authors from five countries and ten research institutions proposed to the *Nuclear Fusion* journal (*NF*) of the International Atomic Energy Agency (IAEA) in Vienna, Austria, a review paper with six sections plus a glossary. This unusually large manuscript had some hundred thousand words and a thousand references. The sections had different lead authors, so that the references of each section were independent of those in other sections, while often repetitive among sections.

The IAEA gave review papers the privilege of waived publication charges (\$150/page), but required authors to ease the publisher's costs by submitting manuscripts of reviews in L^AT_EX, the journal's typesetting system. Therefore, a considerable financial incentive appeared for finding a somewhat automated transformation of all the Word sources into a unified L^AT_EX source.

I was the editor of IAEA's *NF* from mid-1996 to mid-2002 with primary responsibility for the refereeing system and the development of the journal. Previous experience in Unix and L^AT_EX for my own research brought an unofficial role as adviser to the IAEA production office on shell scripts, L^AT_EX, regular expressions, `perl`, and web mounting.

Since the paper appeared valuable from the point of view of journal development, and at the same time a challenge in computer processing, I became particularly interested, and encouraged the authors find ways to satisfy the IAEA requirement: a L^AT_EX manuscript to better support refereeing and eventual publication.

In the end, the paper in question [1] was published in *NF* and was very well received by the research community, at great credit to the co-authors and also good for *NF*.

When the recent call for papers at PracTex came in from Karl Berry, it occurred to me that the story might be interesting for this audience.

Translation from Word to L^AT_EX

At the time of submission (mid-1999) the IAEA and *NF* had investigated with a consultant conversions from Word to L^AT_EX, but had not found a satisfactory solution. One of the twelve authors suggested "`rtf2latex2e`" by Ujwal Setlur Sathyam (now Ujwal Setlur) and Scott Prahl, following the Word-native RTF (Rich Text Format) writer.

According to Microsoft, msdn.microsoft.com,

"The Rich Text Format (RTF) Specification provides a format for text and graphics interchange that can be used with different output devices, operating environments, and operating systems. RTF uses the ANSI, PC-8, Macintosh, or IBM PC character set to control the representation and formatting of a document, both on the screen and in print. With the RTF Specification, documents created under different operating systems and with different software applications can be transferred between those operating systems and applications."

`rtf2latex2e` uses the RTF reader by Paul DuBois and converts RTF files to L^AT_EX2_ε. Some features are: detects text style: (bold, italic, etc.); reads embedded figures; reads tables; converts embedded MathType; converts most Greek and math symbols; reads footnotes; translates hyperlinks. It should compile on any platform that supports a C

compiler. Versions for Macintosh, Unix-type systems, and Windows are available. The distribution, issued under the terms of the GNU General Public License as published by the Free Software Foundation, comes with example `.rtf` files.

The current, and final, version of `rtf2latex2e` can be found at the Comprehensive T_EX Archive Network, ctan.org/tex-archive/support/

`rtf2latex2e`
or at the Source Forge, sourceforge.net.

```
The result of translation gave the expected
\documentclass{article}
\begin{document}
\section*{1. INTRODUCTION}
```

and looked good regarding mathematics and tables, but left all citations as footnotes with the expected chaos with repeated references. A typical reference (of the thousands) appeared many times with different chapter-based numbers.

The footnotes were rendered, for example, as `[1.\footnote{[1.] Author, A., Some Journal \textbf{36} (1997) 123.}]` in section 1, but in generally the same way in section 2, except that the “[1.” became “[2.”.

One task is to transform the `\footnote` style that survives after the Word-rtf- \LaTeX transformation into the normal `\cite{...}-\bibitem{...}` representation of for references. More complicated is to detect as identical those references presented with slight differences; and to detect as distinct references that are actually different but “look” similar.

The goal was a *unique* citation in the body, such as `\cite{AuthorA36p123}`, and a corresponding entry in the bibliography, such as `\bibitem{AuthorA36p123} Author, A., Some Journal, 36 (1997) 123.`

The power of “`perl`” (Practical Extraction and Reporting Language) and its version of “regular expressions” made order from chaos, and produced material suitable for refereeing, and, eventually, publication.

Basic Regular Expressions

A “regular expression” (regex) is a generalized string for matching patterns, and possibly replacing whatever is found found with something else. The programs `grep` (Global Regular Expression Printer) `sed` (Stream EDitor) and the text editor `emacs`, all of which are part of Unix-like systems, incorporate regex-es. (The tools mentioned above had versions workable under Windows 95, but comments on the capability of later Windows and Macintosh systems are outside the scope of this document.)

For a flavor of the regex world:

```
/s/0ld/New/g : Old → New globally (g)
/s/^0ld/New/ : Old → New at line start (^)
/s/(...)0ld/NEW\1/ : xyxOld → NEWxyz
```

In the last example, the string `0ld` is sought, but only if it preceded on its line by 3 characters, which are to be remembered by the parentheses () with the label `\1`. Then, `0ld` is to be replaced by `NEW` but *followed by* the 3 characters just found (here called `xyz`).

These examples only suggest the full power of searching and replacing available, in particular with `perl`.

A short summary of the regex usages is in “Linux in a Nutshell” [2], and an excellent introduction is in the Wikipedia [3].

For an advanced treatment, see “Mastering Regular Expressions” [4]. The “GNU Emacs Manual” [5] explains using `regex-es` in editing.

The prime documentation of `perl` is the “camel book” now in its third edition [6]. The *NF* Office happened to rely mostly on the “llama book” [7] and the pocket-size Desktop Reference by Johan Vromans [8].

Manipulating the References

The lead author of Ref. 1, Gianfranco Federici, contacted a colleague, Andreas Schott, about the challenge of rationalizing the references. Schott produced a `perl` script `foot2cite.pl` which accomplished the task.

A few years previously the *NF* Office had developed a collection of `bash` [9] shell and `perl` scripts to produce print masters and files for mounting pdf and html [10] of articles on the IAEA web server. The \LaTeX source of individual articles led to tables of contents and indexes of authors and subjects from individual article source files with the help of native \LaTeX markup plus additional markup commands of the local style file. ¿From that experience¹ it appeared interesting to develop an IAEA-local program which could be the base of solutions that might be needed in the future. Some features of the resulting `ref_manip.pl` are described in the following.

The idea is to use the “hash” facility in `perl`. Here, a hash is a 1-dimensional array in which the index and the value of the index are both character strings. The 1-to-1 hash `num2cite` connected the Word-original reference number, such as “1.101,” to a string designed to be unique (except in pathological circumstances) such as “`AuthorA36p123.`”

¹ The utility of combining \LaTeX with scripting languages has been explored recently for the TUG; see for example William M. Richter, “T_EX and Scripting Languages” TUGboat, Vol 25, No. 1, p 71 (2004).

For diagnostic purposes the 1-to-1-or-more hash `cite2nums` connected the (uniquely created) `\cite` and `\bibitem` string such as “AuthorA36p123” to the (multiple, in general) original reference numbers such as “1.101”; “2.45”.

The multiple \LaTeX section files produced by `rtf2latex2e` are scanned in sequence for a footnote. If footnote-style text of the nature author-journal-volume-page is found, then an identifier string is made of the first author’s last name, first initial, volume number, page number (AuthorA36p123). The text of the reference is entered in a holder for the bibliography under `\bibitem{AuthorA36p123}`, while the footnote is replaced by `\cite{AuthorA36p123}`. Next, the two hashes receive the appropriate entries — such as “1.101” and “AuthorA36p12” — with the help of a counter in the `perl` script. That counter should not become out of synchronization with the footnote numbers given in the paper unless there is a mistake in the original text.

The references not citing journal articles are detected by the absence of a bolding of an alpha-numeric volume number in a footnote. In that case, the `cite/bibitem` identifier is formed from the first twenty alpha-numeric characters in the citation, excluding all white space. Again, the text of the reference goes to the bibliography as a `\bibitem`.

If the footnote is to a previously used number, such as [1.101] or [2.202], then the `num2cite` hash is used to enter the citation with the `\cite` format, without adding anything to the bibliography.

In the script as developed, pre-processing of the raw section files does, for example, the following:

- take out explicit section numbering
- make all citations (recall, they are of the `\footnote` type) begin in column one as `\CITE[...]` and occupy one entire (sometimes very long) line
- make the bolded volume numbers into a particular form that will not confuse later searches for a right brace closing the footnote.

That pre-processing is no doubt a sign of ignorance of the full power of `perl`, and no doubt extends the execution time. However, execution time is not a practical issue, but being able to construct the script in small pieces that do small, easily testable, things was very much an issue in the environment of the *NF* editorial and production offices.

The final pass changes `\CITE[...]` into `\cite{...}`, writes the `\bibitem{...}` entries, and, optionally, saves the hashes `num2cite` and `cite2nums` for diagnostics.

There are vulnerabilities. A simple one, which could be programmed around, is that the original footnotes cannot contain inside them the characters [] or, other than for volume bolding, { }. A more difficult vulnerability, practically speaking inevitable, is that truly identical references have to be presented in pretty much the same way. There is probably no automated way to defend against typographical errors in the names, volume or page numbers. (The potential vulnerability to different amounts white space had a simple defense.) However, an off-line sort of all the `\cite` and `\bibitem` texts would have a good chance of revealing a problem.

The Result

The processing into \LaTeX of the first draft manuscript created one format completely common to all contributing institutions and authors. With that common form, adjustments in response to the concerns of the *NF* editorial office and referees became easier, as did changes originating with the paper’s authors as the review developed. Even so, the refereeing process was extensive, which is not uncommon for articles appearing in *NF*, and particularly articles of such a length.

Independent of what the authors of Ref. 1 feel about their article and the process of publishing it, the publishing journal and its home organization have interests.

The Institute for Scientific Information (ISI) keeps track of an “Impact Factor” (IF) for thousands of journals [11]. The IF is (at least approximately) the number of citations to a journal divided by the number of articles in the period studied. *Nature* and *Science* have IFs in the 20–30 range. The very prestigious *Physical Review Letters* has an IF around 6, and the *Physical Review*, (series A, B, C, D, and E) is typically between 2 and 3. Journals covering plasma physics and nuclear fusion range from 0.5 to 3 or so, and *NF* is consistently the highest in the group. In the six years ending in 2003 *NF* was between 2.2 and 3.4.

According to Google’s newly introduced “Scholar” service, articles from all journals covered referenced Ref. 1 23 times, which is unusually high for the sub-field of science and engineering covered by *NF*. (The time frame was not apparent from the information at Google.)

Records available at the IAEA show that for Ref. 1: there were 162 downloads in 2003, placing it number 7 in the top 10 downloads for that year; and that the citation rate is roughly double the next most cited article, and far above the average rate.

The numbers quoted above suggest that the research community received Ref. 1 unusually well, making it a fine credit to each of the authors and to their institutions. The numbers also say that the article had significant positive influence on the IF of *NF*, and therefore a positive influence on the continued success of *NF*. In other words, the appearance of this article was very good for its authors as well as the IAEA and *NF*.

Remembering that publication in *NF*, and at low cost to submitters, required a L^AT_EX manuscript, one can wonder if all the good news would have happened without `rtf2latex2e` and `perl`. My speculation:

1. the research paper would have come out, if at all, later than it did,
2. it would not have appeared in *NF*,
3. the authors would not have gotten quite the recognition they did,
4. the IAEA and its journal would have a lower IF for the relevant period.

Acknowledgments

Co-author of `rtf2latex2e`, Ujwal S. Setlur, assisted the co-authors of Ref. 1 during the preparation of a manuscript that the IAEA would accept. As mentioned previously, Andreas Schott, a computer professional experienced in `perl`, produced the script that the co-authors actually used.

L^AT_EX production and web-posting at the IAEA owed particular thanks to M. Bergamini-Rödler, N. Douchev, H. Giller, P. Gillingwater, F. Hannak, I. Kurtev, A. Primes, N. Robertson, M. Sherwin, J. Weil, and, for the L^AT_EX-to-html translations, I. Hutchinson, author of `tth` [10]. The management support of R. Kelleher and D. McLaughlin was indispensable, particularly as *NF* production processes grew to depend heavily on the tools of Unix shell scripts, `perl`, native and locally developed L^AT_EX markup.

In January 2002, the Institute of Physics Publishing (IoPP) of Bristol, UK, assumed responsibility for production (again, based on L^AT_EX) while the IAEA editorial office, located in Vienna, Austria, continued to manage content. The Federici paper [1] is now mounted on the web by the IoPP. The present editor of *NF* is F.C. Schüller of The Netherlands.

David Walden contributed helpful comments on a preliminary draft of this paper.

References

- [1] G. Federici, C. H. Skinner, J. N. Brooks, J. P. Coad, C. Grisolia, A. A. Haasz, A. Hassanein,

V. Philipps, C. S. Pitcher, J. Roth, W. R. Wampler, D. G. Whyte, “Plasma-material interactions in current tokamaks and their implications for next step fusion reactors,” *Nucl. Fusion* **41** No. 12R (2001) 1967-2137.

- [2] Jessica Perry Hekman, “Linux in a Nutshell,” O’Reilly and Associates, Inc., 1997.
- [3] Wikipedia, the Free Encyclopedia, en.wikipedia.org/wiki/Regular_expression
- [4] Jeffrey E. F. Friedl, “Mastering Regular Expressions,” O’Reilly and Associates, Inc., 1997.
- [5] “The GNU Emacs Manual,” 14th edition for version 21.3, Free Software Foundation, 2004. Online at www.gnu.org/software/emacs/manual
- [6] Larry Wall, Tom Christiansen, Jon Orwant, “Programming Perl,” (Third Edition) O’Reilly and Associates, Inc., 2000.
- [7] Randal L. Schwartz and Tom Christiansen, “Learning Perl,” O’Reilly and Associates, Inc., 1997.
- [8] Johan Vromans, “Perl 5 Desktop Reference,” O’Reilly and Associates, Inc., 1996.
- [9] Cameron Newham and Bill Rosenblatt, “Learning the bash Shell,” O’Reilly and Associates, Inc., 1995.
- [10] Ian H. Hutchinson, “TtH: a TeX to HTML translator,” hutchinson.belmont.ma.us/tth/manual/
- [11] See <http://www.isinet.com/>.

Beamer by Example

Andrew Mertz
William Slough

Mathematics and Computer Science Department
Eastern Illinois University

June 15, 2005

Practical TeX 2005

Overview

- ▶ Benefits of Using Beamer
- ▶ Examples
 - ▶ A tiny example
 - ▶ Basic frame ingredients
 - `\begin{frame}`, `\frametitle`, `\end{frame}`
 - ▶ Static frame contents
 - ▶ Lists, mathematics, tables, verbatim text, graphics
 - ▶ Colors and tables via `xcolor`
 - ▶ Two columns
 - ▶ Incremental frame contents
 - ▶ Tables and lists with `\pause`
 - ▶ Lists with `\onslide`
 - ▶ Tic-tac-toe with `\onslide` and multiple graphics files
 - ▶ Highlighting items of a list
- ▶ Ornaments: Fonts and Themes
- ▶ Producing N -up output with `pdfjam` and `pdfpages`
- ▶ Pitfalls
- ▶ References

Practical TeX 2005

Benefits of Using Beamer

- ▶ L^AT_EX-based, platform-independent
- ▶ Extensively documented
- ▶ Provides color
- ▶ Rich, dynamic effects
- ▶ Generates PDF output suitable for both presentation and printing
- ▶ Easy to learn and use
- ▶ Flexible

Practical TeX 2005

A Tiny Example

```
\documentclass{beamer}

\title{A Tiny Example}
\author{Andrew Mertz and William Slough}
\date{June 15, 2005}

\begin{document}

\maketitle

\begin{frame}
\frametitle{First Slide}
Contents of the first slide
\end{frame}

\begin{frame}
\frametitle{Second Slide}
Contents of the second slide
\end{frame}

\end{document}
```

Practical TeX 2005

A Tiny Example

Andrew Mertz and William Slough

June 15, 2005

Practical TeX 2005

First Slide

Contents of the first slide

Practical TeX 2005

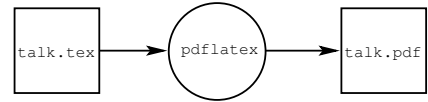
Second Slide

Contents of the second slide



Practical T_EX 2005

Processing



Practical T_EX 2005

What Goes into a Frame?

- ▶ Headline and footline
- ▶ Left and right sidebar
- ▶ Navigation bars
- ▶ Logo
- ▶ Frame title
- ▶ Background
- ▶ Content

Practical T_EX 2005

Frame Content

- ▶ Lists
- ▶ Mathematics
- ▶ Tables
- ▶ Verbatim text
- ▶ Graphics
- ▶ Other packages
- ▶ ...

Practical T_EX 2005

Example: Lists

```
\begin{frame}
\frametitle{Frame Content}
\begin{itemize}
\item Lists
\item Mathematics
\item Tables
\item Verbatim text
\item Graphics
\item Other packages
\item $\ldots$
\end{itemize}
\end{frame}
```

Practical T_EX 2005

L'Hôpital's Rule

If

f and g are differentiable,

$$\lim_{x \rightarrow \infty} f(x) = \infty, \text{ and}$$

$$\lim_{x \rightarrow \infty} g(x) = \infty,$$

then

$$\lim_{x \rightarrow \infty} \frac{f(x)}{g(x)} = \lim_{x \rightarrow \infty} \frac{f'(x)}{g'(x)}.$$

Practical T_EX 2005

Example: Mathematics

```
\begin{frame}
\frametitle{L'H\^{o}pital's Rule}

If
\begin{itemize}
\item[]  $f$  and  $g$  are differentiable,
\item[]  $\lim_{x \rightarrow \infty} f(x) = \infty$ , and
\item[]  $\lim_{x \rightarrow \infty} g(x) = \infty$ ,
\end{itemize}
then
\begin{itemize}
\item[]  $\lim_{x \rightarrow \infty} \frac{f(x)}{g(x)} = \lim_{x \rightarrow \infty} \frac{f'(x)}{g'(x)}$ .
\end{itemize}

\end{frame}
```

Practical TeX 2005

Practical TeX 2005 Events

8-9 am	Registration	
9 am	Karl Berry	Opening
9:15 am	Nelson Beebe	Keynote address
10:15 am	Break	
10:30 am	Peter Flom	A True Beginner Looks at L ^A T _E X
11 am	Anita Schwartz	The Art of L ^A T _E X Problem Solving
11:45 am	Steve Peter	Introduction to memoir
12:30 pm	Lunch	

Practical TeX 2005

Example: Tables

```
\begin{frame}
\frametitle{Practical TeX 2005 Events}

\begin{center}
\begin{tabular}{|r|l|l|}\hline
8-9 am & Registration & \\
9 am & Karl Berry & Opening \\
9:15 am & Nelson Beebe & Keynote address \\
10:15 am & Break & \\
10:30 am & Peter Flom & A True Beginner Looks at LaTeX \\
11 am & Anita Schwartz & The Art of LaTeX Problem Solving \\
11:45 am & Steve Peter & Introduction to memoir \\
12:30 pm & Lunch & \\
\hline
\end{tabular}
\end{center}

\end{frame}
```

Practical TeX 2005

Verbatim Text

```
\begin{frame}
\frametitle{Practical TeX 2005 Events}
\begin{center}
\begin{tabular}{|r|l|l|}\hline
8-9 am & Registration & \\
9 am & Karl Berry & Opening \\
9:15 am & Nelson Beebe & Keynote address \\
10:15 am & Break & \\
10:30 am & Peter Flom & A True Beginner Looks at LaTeX \\
11 am & Anita Schwartz & The art of LaTeX Problem Solving \\
11:45 am & Steve Peter & Introduction to memoir \\
12:30 pm & Lunch & \\
\hline
\end{tabular}
\end{center}
\end{frame}
```

Practical TeX 2005

Example: Verbatim

```
\begin{frame}[fragile]
\frametitle{Verbatim Text}
{\scriptsize
\begin{verbatim}
\begin{frame}
\frametitle{Practical TeX 2005 Events}
\begin{center}
\begin{tabular}{|r|l|l|}\hline
8-9 am & Registration & \\
9 am & Karl Berry & Opening \\
9:15 am & Nelson Beebe & Keynote address \\
10:15 am & Break & \\
10:30 am & Peter Flom & A True Beginner Looks at LaTeX \\
11 am & Anita Schwartz & The Art of LaTeX Problem Solving \\
11:45 am & Steve Peter & Introduction to memoir \\
12:30 pm & Lunch & \\
\hline
\end{tabular}
\end{center}
\end{frame}
}
\end{verbatim}
\end{frame}
```

Practical TeX 2005

Practical TeX 2005 Logo



Practical TeX 2005

Example: Graphics

```
\begin{frame}
\frametitle{Practical \TeX\ 2005 Logo}

\begin{center}
\includegraphics[height=3.25in]{p2005}
\end{center}

\end{frame}
```

Practical \TeX 2005

Colors via xcolor

Practical \TeX
Practical \TeX
Practical \TeX
Practical \TeX
Practical \TeX
Practical \TeX
Practical \TeX
Practical \TeX
Practical \TeX
Practical \TeX

Practical \TeX 2005

Example: xcolor

```
\begin{frame}
\frametitle{Colors via {\color{BrickRed} xcolor}}

\begin{center}
{\color{BlueViolet!10} Practical} {\color{BrickRed!10} \TeX} \\
{\color{BlueViolet!20} Practical} {\color{BrickRed!20} \TeX} \\
{\color{BlueViolet!30} Practical} {\color{BrickRed!30} \TeX} \\
{\color{BlueViolet!40} Practical} {\color{BrickRed!40} \TeX} \\
{\color{BlueViolet!50} Practical} {\color{BrickRed!50} \TeX} \\
{\color{BlueViolet!60} Practical} {\color{BrickRed!60} \TeX} \\
{\color{BlueViolet!70} Practical} {\color{BrickRed!70} \TeX} \\
{\color{BlueViolet!80} Practical} {\color{BrickRed!80} \TeX} \\
{\color{BlueViolet!90} Practical} {\color{BrickRed!90} \TeX} \\
{\color{BlueViolet!100} Practical} {\color{BrickRed!100} \TeX} \\
\end{center}

\end{frame}
```

Practical \TeX 2005

Practical \TeX 2005 Events

8-9 am	Registration	
9 am	Karl Berry	Opening
9:15 am	Nelson Beebe	Keynote address
10:15 am	Break	
10:30 am	Peter Flom	A True Beginner Looks at \LaTeX
11 am	Anita Schwartz	The Art of \LaTeX Problem Solving
11:45 am	Steve Peter	Introduction to memoir
12:30 pm	Lunch	

Practical \TeX 2005

Example: Table with Color

```
\begin{frame}
\frametitle{Practical \TeX\ 2005 Events}

\begin{center}
\rowcolors{1}{RoyalBlue!20}{RoyalBlue!5}
\begin{tabular}{|r|l|l|}\hline
8-9 am & Registration & \\
9 am & Karl Berry & Opening \\
9:15 am & Nelson Beebe & Keynote address \\
10:15 am & Break & \\
10:30 am & Peter Flom & A True Beginner Looks at \LaTeX \\
11 am & Anita Schwartz & The Art of \LaTeX Problem Solving \\
11:45 am & Steve Peter & Introduction to memoir \\
12:30 pm & Lunch & \\
\end{tabular}
\end{center}

\end{frame}
```

Practical \TeX 2005

Revising documentclass for xcolor

```
\documentclass[xcolor=pdftex,dvipsnames,table]{beamer}
```

- ▶ Beamer loads some packages automatically
- ▶ `\usepackage{...}` cannot be used for such packages
- ▶ Package options are specified within documentclass
- ▶ `xcolor=...` specifies options for xcolor
- ▶ `pdftex` specifies the correct driver for pdflatex
- ▶ `dvipsnames` loads the Crayola/dvips colors
- ▶ `table` loads the colortbl package

Practical \TeX 2005

A Race Condition

Two people share a checking account. They each visit an ATM machine in different parts of town at approximately noon. One deposits \$100; the other withdraws \$100.

What happens?

```
Process A                                Process B
...
balance += 100;                          balance -= 100;
...

```

Practical TeX 2005

Example: Two Columns

Two people share a checking account. They each visit an ATM machine in different parts of town at approximately noon. One deposits \$100; the other withdraws \$100.

```
\vspace{2ex} What happens? \vspace{2ex}

\begin{columns}[t]
\column{3cm}
{\color{BrickRed} \centerline{Process A}
\begin{verbatim}
...
balance += 100;
...
\end{verbatim}
}
\column{3cm}
{\color{BlueViolet} \centerline{Process B}
\begin{verbatim}
...
balance -= 100;
...
\end{verbatim}
}
\end{columns}

```

Practical TeX 2005

Incremental Frame Contents

- ▶ Tables with `\pause`
- ▶ Lists with `\pause`
- ▶ Lists with `\onslide`
- ▶ Tic-tac-toe via `tabular` and `\onslide`
- ▶ Tic-tac-toe via multiple graphics files
- ▶ Highlighting items

Practical TeX 2005

Software Complexity

Date	Developer	OS	Lines of C
1976	Thompson/Ritchie	AT&T UNIX	9,000
1997	Tanenbaum	Minix	62,000
1999	Torvalds	Linux	1,000,000
2000	Cutler <i>et al.</i>	Windows NT	28,000,000

Practical TeX 2005

Example: Table with pause

```
\begin{frame}
\frametitle{Software Complexity}

\setbeamercovered{dynamic}
\renewcommand{\arraystretch}{1.2}
\begin{tabular}{c|c|c|c|c}
Date & Developer & OS & Lines of C & U's \\ \hline \hline
1976 & Thompson/Ritchie & AT&T UNIX & 9,000 & \\ \hline \pause
1997 & Tanenbaum & Minix & 62,000 & \\ \hline \pause
1999 & Torvalds & Linux & 1,000,000 & \\ \hline \pause
2000 & Cutler {em et al.} & Windows NT & 28,000,000 & \\ \hline \hline
\end{tabular}

\end{frame}

```

Practical TeX 2005

Frames and Overlays

- ▶ A frame consists of one or more **overlays**
- ▶ Dynamic effects can be achieved by using overlays
- ▶ Inserting `\pause` is one way to create a new overlay

Practical TeX 2005

Possible Objections To Using T_EX

- ▶ I've heard T_EX and L^AT_EX are not WYSIWYG
T_EX is a mark-up language
- ▶ L^AT_EX is not installed on my computer
ProText, T_EXLive, T_EXShop
- ▶ I don't know who to ask for help
T_EX Users Group – www.tug.org
News group – comp.text.tex
Google search

Practical T_EX 2005

Example: List with pause

```
\begin{frame}[fragile]
\frametitle{Possible Objections To Using \TeX}
\setbeamercovered{invisible}
\begin{itemize}
\item I've heard \TeX and \LaTeX are not WYSIWYG\ \pause
{\color{BrickRed} \TeX is a mark-up language} \pause

\item \LaTeX is not installed on my computer\ \pause
{\color{BrickRed} ProText, \TeX Live, \TeX Shop} \pause

\item I don't know who to ask for help\ \pause
{\color{BrickRed} \TeX Users Group -- {\tt www.tug.org}\ \
News group -- {\tt comp.text.tex}\ \
Google search}
\end{itemize}
\end{frame}
```

Practical T_EX 2005

Possible Objections To Using T_EX

- ▶ I've heard T_EX and L^AT_EX are not WYSIWYG
T_EX is a mark-up language
- ▶ L^AT_EX is not installed on my computer
ProText, T_EXLive, T_EXShop
- ▶ I don't know who to ask for help
T_EX Users Group – www.tug.org
News group – comp.text.tex
Google search

Practical T_EX 2005

Example: List with onslide

```
\begin{frame}[fragile]
\frametitle{Possible Objections To Using \TeX}
\begin{itemize}
\item I've heard \TeX and \LaTeX are not WYSIWYG\ \
\onslide<2-> {{\color{BrickRed} \TeX is a mark-up language}}

\item \LaTeX is not installed on my computer\ \
\onslide<3-> {{\color{BrickRed} ProText, \TeX Live, \TeX Shop}}

\item I don't know who to ask for help\ \
\onslide<4-> {{\color{BrickRed} \TeX Users Group -- {\tt www.tug.org}\ \
News group -- {\tt comp.text.tex}\ \
Google search}}
\end{itemize}
\end{frame}
```

Practical T_EX 2005

Overlay Specifications

- ▶ Overlays are numbered 1, 2, 3, ..., *N*
- ▶ An **overlay specification** defines a sequence of numbers
- ▶ Examples:
 - ▶ <2>
 - ▶ <2-4>
 - ▶ <2->
 - ▶ <1,3,5,7,9>
 - ▶ <-3,8->
- ▶ Some L^AT_EX commands can provide an overlay specification
 - ▶ `\textbf<2->{bolded text}`
 - ▶ `\emph<3,5>{emphasized text}`
 - ▶ `{\color<4>{RoyalBlue} blue}`

Practical T_EX 2005

Tic-Tac-Toe via tabular

O	X	X
X	O	O
X	O	X

Practical T_EX 2005

Example: Tic-Tac-Toe via tabular

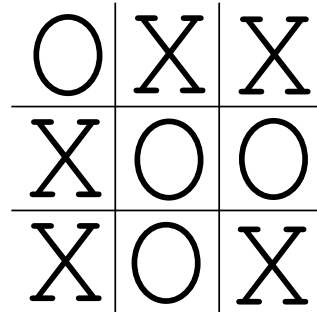
```
\begin{frame}
\frametitle{Tic-Tac-Toe via {\tt tabular}}

\setbeamercovered{invisible}
{\Huge
\begin{center}
\begin{tabular}{c|c|c}
\onslide<9->{0} & \onslide<8->{X} & \onslide<2->{X} \\ \hline
\onslide<6->{X} & \onslide<3->{0} & \onslide<5->{0} \\ \hline
\onslide<10->{X} & \onslide<7->{0} & \onslide<4->{X}
\end{tabular}
\end{center}
}

\end{frame}
```

Practical T_EX 2005

Tic-Tac-Toe via Multiple Graphics Files



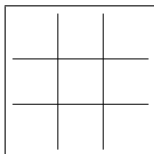
Practical T_EX 2005

Example: Multiple Graphics Files

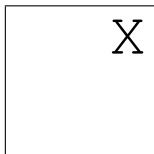
```
\usepackage{xmpmulti}
...
\begin{frame}
\frametitle{Tic-Tac-Toe via Multiple Graphics Files}

\setbeamercovered{invisible}
\begin{center}
\multiinclude[format=pdf,width=3in]{game}
\end{center}

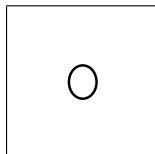
\end{frame}
```



game-0.pdf



game-1.pdf



game-2.pdf

Practical T_EX 2005

Highlighting Items of a List

- ▶ Practical
- ▶ T_EX
- ▶ 2005

Practical T_EX 2005

Example: Multiple Overlay Specifications

```
\begin{frame}
\frametitle{Highlighting Items of a List}

\setbeamercovered{dynamic}
\begin{itemize}
\item<1-> {\color<1>{BrickRed} Practical}
\item<2-> {\color<2>{BrickRed} \TeX}
\item<3-> {\color<3>{BrickRed} 2005}
\end{itemize}

\end{frame}
```

Practical T_EX 2005

Highlighting Items of A List Revisited

- ▶ Practical
- ▶ T_EX
- ▶ 2005

Practical T_EX 2005

Example: Default Overlay Specifications

```
\begin{frame}
\frametitle{Highlighting Items of A List Revisited}

\setbeamercovered{dynamic}
\setbeamercolor{alerted text}{fg=BrickRed}
\begin{itemize}[<- | alerted@>]
\item Practical
\item \TeX
\item 2005
\end{itemize}

\end{frame}
```

Practical \TeX 2005

Other Fonts

- ▶ Other fonts installed with \TeX can be used
- ▶ For example, to use the Microsoft Comic Sans font, place

```
\usepackage{comicsans}
\renewcommand{\sfddefault}{comic}
```

in the preamble

Practical \TeX 2005

Beamer by Example

Andrew Mertz
William Slough

Mathematics and Computer Science Department
Eastern Illinois University

June 15, 2005

Practical \TeX 2005

Practical \TeX 2005

Overview

- ▶ Benefits of Using Beamer
- ▶ Examples
 - ▶ A tiny example
 - ▶ Basic frame ingredients

```
\begin{frame}, \frametitle, \end{frame}
```
 - ▶ Static frame contents
 - ▶ Lists, mathematics, tables, verbatim text, graphics
 - ▶ Colors and tables via `xcolor`
 - ▶ Two columns
 - ▶ Incremental frame contents
 - ▶ Tables and lists with `\pause`
 - ▶ Lists with `\onslide`
 - ▶ Tic-tac-toe with `\onslide` and multiple graphics files
 - ▶ Highlighting items of a list
- ▶ Ornaments: Fonts and Themes
- ▶ Producing N -up output with `pdfjam` and `pdfpages`
- ▶ Pitfalls
- ▶ References

Practical \TeX 2005

Practical \TeX 2005

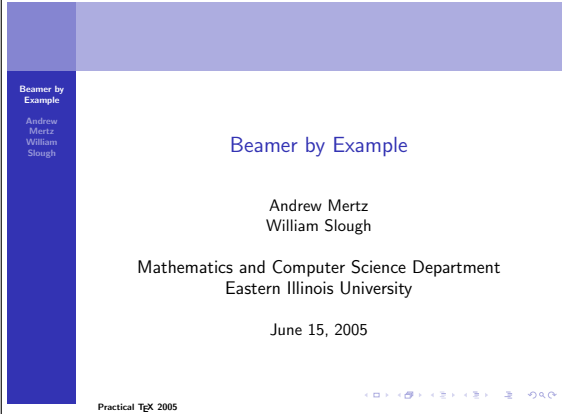
Other Themes

- ▶ The Beamer package supplies many themes
- ▶ For example, place

```
\usepackage{beamerthemesplit}
\usetheme{Berkeley}
\usecolortheme{dolphin}
```

in the preamble

Practical \TeX 2005



The slide features a blue vertical bar on the left side with the text "Beamer by Example", "Andrew Mertz", and "William Slough". The main content area has a white background with a blue header bar at the top. The text "Beamer by Example" is centered at the top, followed by the authors' names, the department and university, and the date "June 15, 2005". Navigation icons are at the bottom right.

Practical \TeX 2005

Overview

Beamer by Example
Andrew Mertz, William Slough

- Benefits of Using Beamer
- Examples
 - A tiny example
 - Basic frame ingredients


```
\begin{frame}, \frametitle, \end{frame}
```
 - Static frame contents
 - Lists, mathematics, tables, verbatim text, graphics
 - Colors and tables via `xcolor`
 - Two columns
 - Incremental frame contents
 - Tables and lists with `\pause`
 - Lists with `\onslide`
 - Tic-tac-toe with `\onslide` and multiple graphics files
 - Highlighting items of a list
- Ornaments: Fonts and Themes
- Producing *N*-up output with `pdfjam` and `pdfpages`
- Pitfalls
- References

Practical TeX 2005

Practical TeX 2005

pdfjam and pdfpages

- ▶ Beamer output can be created without overlay effects:


```
\documentclass[handout,xcolor=pdftex,dvipsnames,table]{beamer}
```
- ▶ The `pdfpages` package can be used to generate an *N*-up version from an overlay-free presentation
- ▶ `pdfjam` is a Unix shell script to automate the use of `pdfpages`

Practical TeX 2005

Pitfalls

- ▶ Beamer's `verbatim` environment
- ▶ Multiple processing steps
- ▶ Switching themes after content is established
- ▶ Frame numbering
- ▶ Temptation to exert fine control

Practical TeX 2005

Credits and Acknowledgments

<code>beamer</code>	Till Tantau
<code>mpmulti</code>	Klaus Guntermann
<code>pdfjam</code>	David Firth
<code>pdfpages</code>	Andreas Matthias
<code>xcolor</code>	Uwe Kern

Practical TeX 2005

References

- ▶ Beamer: latex-beamer.sourceforge.net
- ▶ TeX Live: www.tug.org/texlive/
- ▶ TeX Shop: www.uoregon.edu/~koch/texshop/texshop.html
- ▶ ProTeXt: www.tug.org/protext/
- ▶ `pdfpages`: www.ctan.org/tex-archive/help/Catalogue/entries/pdfpages.html
- ▶ `pdfjam`: www2.warwick.ac.uk/fac/sci/statistics/staff/academic/firth/software/pdfjam

Practical TeX 2005

NO MATTER IF
YOUR DOCUMENTS
ARE

BIZARRE

smokin'

friendly

FORMAL

OR JUST A LITTLE

$$\frac{\text{odd}}{2} \neq \left\lfloor \frac{\text{odd}}{2} \right\rfloor$$

THEY'RE SPECIAL
TO YOU

SO BRING THEM
TO US

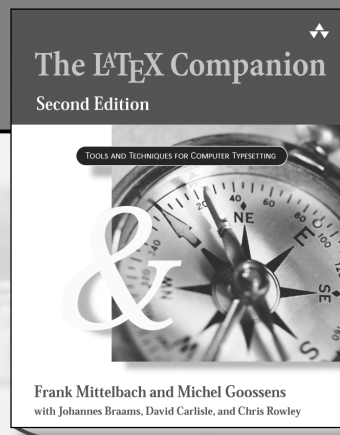
silmaril
consultants

- ▷ L^AT_EX typesetting
- ▷ XML document systems

<http://www.silmaril.ie/>

The L^AT_EX Companion

Second Edition



ISBN: 0-201-36299-6

Frank Mittelbach and Michel Goossens
*with Johannes Braams,
David Carlisle, and Chris Rowley*

The L^AT_EX Companion has long been the essential resource for anyone using L^AT_EX to create high-quality printed documents. This completely updated edition brings you all the latest information about L^AT_EX and the vast range of add-on packages now available—over 200 are covered. Like its predecessor, *The L^AT_EX Companion, Second Edition* is an indispensable reference for anyone wishing to use L^AT_EX productively.

Available at fine bookstores everywhere.


Addison
Wesley

For more information, visit:
[www.awprofessional.com/
titles/0201362996](http://www.awprofessional.com/titles/0201362996)

Entertainment Tax Benefits
Health and Happiness
Creativity

Helping Buyers and Sellers Succeed

Joe Hogg, Broker-Associate
Seniors Real Estate Specialist; Realtor
 **Tri-City**, Los Angeles, CA 90027

voice: (323) 842-9764
email: joe@joehogg.com
web: <http://www.joehogg.com>

Building Equity

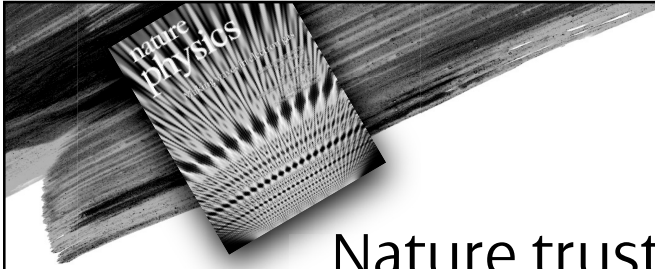
Do you need on-site training for **ETEX**?

Contact Cheryl Ponchin at

cponchin@comcast.net

Training will be customized for your company needs.

Any level, from Beginning to Advanced.



Nature trusts us... You can too!

Nature Physics is a new journal. Like others from Nature Publishing Group, it is set to become the leader in its field.

And it'll be typeset using T_EX. And of course River Valley were chosen for the task. Needless to say, the final PDF will come from XML with zero intervention.

See why NPG, IOP and Elsevier trust us with their toughest journals. Perhaps we can breath new life into **your** journals.

www.river-valley.com

ScientificWorkPlace® ScientificWord®

Mathematical Word Processing • L^AT_EX Typesetting • Computer Algebra

Version 5 Sharing Your Work Is Easier

- ◆ Typeset PDF in the only software that allows you to transform L^AT_EX files to PDF, fully hyperlinked and with embedded graphics in over 50 formats
- ◆ Export documents as RTF with editable mathematics (Microsoft Word and MathType compatible)
- ◆ Share documents on the web as HTML with mathematics as MathML or graphics

The Gold Standard for Mathematical Publishing

ScientificWorkPlace makes writing, sharing, and doing mathematics easier. A click of a button allows you to typeset your documents in L^AT_EX. And, you can compute and plot solutions with the integrated computer algebra engine, *MuPAD*® 2.5.



Email: info@mackichan.com • Toll-free: 877-724-9673 • Phone: 360-394-6033
Visit our website for free trial versions of all our software.

www.mackichan.com/tug

In the Monte Carlo simulations that follow, three bandwidth choices are parameter combination: The LSCV bandwidth, the "Stanton" bandwidth, and independent and identically distributed (IID) bandwidth. The first choice is the least squares cross validation problem (ref. LSCVfunc). The IID bandwidth for IID data, and it is defined as $h^{iid} = \hat{\sigma} T^{-1/5}$, where $\hat{\sigma}$ is the sample standard deviation and T is the sample size. The Stanton bandwidth is the one actually used in Stanton (1997) are based directly on equations (ref. Stanton 1997). In particular, "inverting" these equations yields:

$$\mu(x_i) = \frac{1}{\Delta} E[x_{i+\Delta} - x_i | x_i] + \frac{o(\Delta)}{\Delta}$$

$$\sigma(x_i) = \sqrt{E[(x_{i+\Delta} - x_i)^2 | x_i] \frac{1}{\Delta} + \frac{o(\Delta)}{\Delta}}$$

The essence of Stanton's approach is to apply the Nadaraya-Watson (Nadaraya-Watson) regression estimator to construct nonparametric estimates of the conditional mean (ref. diff2) and (ref. diff2):

$$\hat{\mu}(x_i) = \frac{1}{n} \sum_{j=1}^{n-1} (x_{j+1}^\Delta - x_j^\Delta) K\left(\frac{x_i - x_j^\Delta}{h}\right)$$

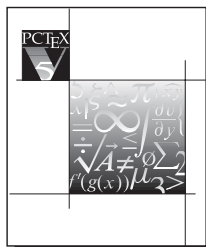
Screen text is reprinted from an article in the Journal of Finance.



PCT_EX v5 Makes Using L^AT_EX a Snap!

PCT_EX v5 for Windows has new, enhanced features in an integrated user interface to compose L^AT_EX and T_EX documents:

- ▼ Smooth edit–typeset–preview workflow
- ▼ Project organizer
- ▼ PostScript preview
- ▼ Built-in PDF distiller
- ▼ Export to Adobe Illustrator
- ... and more!



Pricing starts at \$99, complete system including MathTimeProfessional fonts is \$399. Upgrades from earlier PCT_EX versions start at \$29.

PCT_EX v5 also offers

The MathTimeProfessional Fonts

Mathematics symbols to match the Times fonts,

$$\alpha\beta\gamma\delta \pm \zeta\kappa\xi\psi\omega \mp \partial f(x, y, z)/\partial x$$

with *individually designed fonts* for different sizes

$$\alpha\beta\gamma\delta\Gamma\Delta\Theta \dots \alpha\beta\gamma\delta\Gamma\Delta\Theta \dots \alpha\beta\gamma\delta\Gamma\Delta\Theta \dots$$

to provide more readable symbols in superscripts

$$\sqrt{\frac{x^y + \alpha^{\beta\gamma} + \Theta^{\Delta\Gamma} + f_{\alpha\beta}(x_{ij}^{\lambda^p}, y_{ij}^{\mu^q}, z_{ij}^{\nu^r})}{\epsilon\eta\theta + i\kappa\lambda + \pi\rho\sigma + (\vartheta\varpi\phi + \Xi\Phi\Omega)^{\phi\chi\epsilon + \Psi}}}$$

And: parentheses and radicals up to **4 inches high** and math accents extending up to **4 inches wide!**

With easy to use guide for L^AT_EX and plain T_EX



For more information and for a Free 30-day evaluation, visit www.pctex.com
 Personal T_EX, Inc. 800-808-7906 415-296-7550 sales@pctex.com

Math and Science Equations 40% Faster!

MathType for Windows and Macintosh

MathType is the full-featured, professional version of the Equation Editor in Microsoft Office. It has hundreds of extras that you do not get with Equation Editor, and installs special features into Microsoft Word/PowerPoint, that effectively creates an integrated math word processor, slide creator, and web page editor.

Get the job done faster!

Tests have proven that using MathType gets the same job done 40% faster than using Equation Editor.

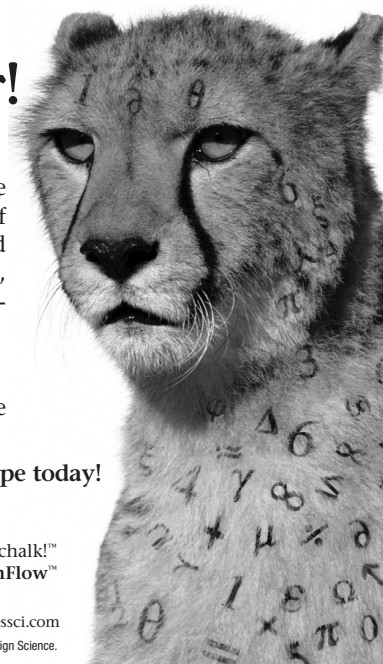
Download a fully-functional 30-day trial of MathType today!
www.dessci.com



MathType™ The best thing for writing equations since chalk!™
 MathType™ WebEQ™ MathPlayer™ MathFlow™

Design Science, Inc. 4028 Broadway, Long Beach, CA 90803, USA
www.dessci.com Telephone: 562-433-0685, Fax: 562-433-6969, Email: sales@dessci.com

MathType, "The best thing for writing equations since chalk!" and "How Science Communicates" are trademarks of Design Science. All other company and product names are trademarks and/or registered trademarks of their respective owners.



Design Science
 How Science Communicates™