ABSTRACTS

1 Monday, August 16, 1999
TEX and Math on the Web

1.1 TEX and the Web in the Higher Education of the Future: Dreams and Difficulties

Stephen A. Fulling
Texas A&M University
fulling@math.tamu.edu

Abstract: New technology provides an opportunity to move mathematical and technical education out of the lecture-homework-test mold into modes that do more to develop students' communication skills, teamwork, attention to quality, and overall responsible, mature behavior.

In practice, however, severe and presumably unnecessary obstacles are encountered, mostly connected with the difficulty of transmitting mathematical notation electronically. I will describe a partially successful scheme for carrying out peer review of homework papers over e-mail and the Web, resulting in a student-written solutions manual, and will offer a wish list of technical improvements.

1.2 MathML: The Key to Math on the Web

Patrick D.F. Ion
Mathematical Reviews / AMS
Co-Chair W3C Math Working Group
ion@ams.org

Abstract: As the Web gains in importance and as the needs of mathematical formalism on the Web are beginning to be met by MathML, it is an opportune time to reflect on the design decisions made by the W3C Math Working Group that resulted in the verbose markup language for transport of math on the Web that MathML turns out to be. The TEX community need not be frightened by the advent of MathML, but may learn to work in the Web environment it provides.

The presentation will describe some of the key aspects of MathML, and explain how it has begun to be used (by August 1999 the support for MathML may be expected to be much greater in a practical sense than it is now). Expected future developments and roles will be commented upon.

1.3 TEXML brings TEX to Web Future

Doug Lovell
IBM Research, New York
dcl@us.ibm.com

Abstract: XML, eXtensible Markup Language, is a simplified subset of SGML, which is fast becoming a standard for content management on the internet.

TEXML is an XML vocabulary for TEX. A processor written in JAVA translates TEXML-conforming XML into TEX. The processor provides a document formatting solution for XML that leverages the rich knowledge and capability built over many years in TEX.

The presentation describes the TEXML document format and the processor, TEXMLatte, that produces TEX source from TEXML markup.

1.4 Using MathType to Create TEX and MathML Equations

Paul R. Topping
Design Science, Inc., California
pault@mathtype.com

Abstract: This presentation starts with an overview of the capabilities of MathType 4.0, the latest release of Design Science's interactive mathematical equation editing software package for the Apple Macintosh and Microsoft Windows platforms. MathType is the full-featured version of Design Science's Equation Editor software, distributed with Microsoft Office, Word, and Works, Corel WordPerfect, and many other software products.

MathType has a simple direct-manipulation interface for creating standard mathematical notation. It is not a word processor but is used to create or edit mathematical expressions one at a time to be inserted into a word processing or TEX document. Instead of entering a computer language, such as TEX, the MathType user combines simple typing with the insertion of "templates". For example, inserting a fraction template results in a fraction bar with empty slots above and below for the numerator and denominator. The contents of each slot can be completed by the user by more typing and inserting of templates. The displayed equation is reformatted as the user types and spacing is added automatically (although spacing may be explicitly overridden).

This interface is simpler than direct TEX input as there are no keywords to remember and, most importantly, no possibility of syntax errors. The interface is also faster as keystroke shortcuts can be assigned to all symbols and templates. Common sub-expressions can be saved by the user, assigned a keystroke or a toolbar button, to be recalled at any point in equation construction with a keystroke or mouse-click. And, because MathType is UNICODE-based, it can handle non-English language text with ease.

While MathType has had a TEX translator for many years, until version 4.0, it had two important limitations: it could only generate plain TEX, and the user had no control over the TEX fragments generated for particular symbols and templates. MathType 4.0 features a complete re-design of the translator mechanism.

The translation of a MathType expression is controlled by a translator definition file. Translation of characters is performed via rules that are selected by UNICODE value, making the translator mechanism especially powerful for generating TEX for non-English environments. Although the chief motivation for its development was TEX translation, it can be used to generate other mathematical languages.

MathType is supplied with four TEX translators (plain TEX, AMSTEX, LATEX, and AMS-LATEX) and several MathML translators. These can be customized for specific applications; or, translators for other mathematical languages can be written by starting from scratch. Also, commands are available in Microsoft Word that will allow a Word document containing MathType (or Equation Editor) equations to be converted to TEX or any other language supported by a translator. MathType's translation facilities can be used as an aid to learning TEX, as a simpler interface for entering equations into a TEX authoring system, or as part of a document conversion scheme for journal and book publishers.

1.5 LaTeX to XML/MathML (workshop)

Eitan Gurari
Ohio State University
gurari@cis.ohio-state.edu

and

Sebastian Rahtz
Oxford University, UK
sebastian.rahtz@oucs.ox.ac.uk

Prerequisites: Some understanding of the principles of XML.

Description: This workshop will discuss the translation of LaTeX to XML/MathML on an abstract level, and will show how TeX4ht can be used for the implementation. In-depth coverage of these topics can be found in the forthcoming book *The LaTeX Web Companion*, written by Michel Goosens and Sebastian Rahtz, with Eitan M. Gurari, Ross Moore, and Robert S. Sutor (Addison Wesley Longman, Summer 1999).

1.6 Models and Languages for Formatted Documents

Chris Rowley
Open University, UK
C.A.Rowley@open.ac.uk

Abstract: This presentation will investigate some issues related to the evolution of TEX's standard output format for future extensions of TEX.

Many of my thoughts about such a future standard output format are informed by the needs of documents that are designed (to high standards) for multiple uses, in particular for reading both on screen and in paper form.

Many people now have experience of using TEX as the typesetting engine for such documents where the multi-use output form is a 'PDF document' — PDF here refers to the Adobe's Portable Document Format.

Although PDF is technically not a Device-Independent language, it does contain a large core of stuff that is, at least potentially, as 'device independent' as TEX's DVI language. Both must, of course, be parsed by an application that understands the language and its underlying document- and page-models; and although they look very different at the detailed level, their page-models and abstract semantics also have a lot in common. This is why the part of pdfTEX that handles classic TEXfiles is only very locally and minimally different from classic TEX.

However, PDF has a far richer document model, and hence pdfTEX has a number of extra primitives. Many of these (such as those which support hyper-links and pop-ups) clearly implement particular abstractions that are (as seen by modern practitioners) essential parts of on-screen documents, whilst others (such as thumb-nails) are peculiar to the somewhat old-fashioned and arbitrary interface to document structure chosen by the designers of various Adobe products. There probably also exist necessary extensions to the currently used models that are not supported well by either language.

By August 1999 I hope, with a bit of help from my friends, to have further analysed the models and concepts that need to be supported for multi-use documents and how well PDF does this. I am confident that this will prove useful in considering in more detail the necessary extensions of TEX's DVI standard.

1.7 AcroTEX: Acrobat and TEX Team Up

D. P. Story
University of Akron, Ohio
dpstory@uakron.edu

Abstract: Adobe's Acrobat (PDF) and Donald Knuth's TEX system make a powerful team for putting mathematics on the internet. For the educator, this team, called "AcroTEX," is the poor man's multimedia software company.

Though TEX was implemented before the rise in popularity of the WWW, PostScript code written to the DVI file, using TEX's `\specials`, can be used to enhance an electronic document created from a TEX source by introducing such elements as color, hypertext links, form features, sounds, and even video clips. These special features are achieved by inserting 'pdfmarks' into the output file. The Adobe Distiller, in turn, interprets these pdfmarks and translates them into the appropriate element as it writes the PDF document. TEX is, therefore, well suited for creating PDF files, especially technical material. With the aid of the very powerful macro facility and the ability to position material very precisely on a page, these electronic enhancements can be created and placed in an exact and automated way.

The talk explores the capabilities of AcroTEX and the contents of the AcroTEX web site (http://www.math.uakron.edu/ dpstory); examples include tutorials, an electronic grading system, mathematical games, and technical articles.

1.8 Using LaTeX to Create Quality Interactive PDF Documents for the WWW (workshop)

D. P. Story
University of Akron, Ohio
dpstory@uakron.edu

Prerequisites: Some experience with LaTeX.

Description: How to create quality interactive PDF documents for the WWW using LaTeX. Topics include:

- how to set up the Acrobat/TeX System. A Win95 based presentation
- page layout for a PDF document, designed to be read over the web
- how to add a dash of color to your text and background
- use of `Hyperref`, a package by Sebastian Rahtz, and its "∞-many" options — only finitely many will be discussed, however.

Macros will be introduced on how to:

(1) create problem exercises with hyperlinked solutions;

(2) create multiple choice questions with instant feedback;

(3) create multiple choice graded quizzes using JavaScript.

2 Tuesday August 17, 1999
Customizing Document Layout

2.1 Doing It My Way: A Lone TEXer in the Real World

Jean-luc Doumont
JL Consulting, Brussels, Belgium
JL@JLConsulting.be

Abstract: While a world-renowned standard in many academic fields, Don Knuth's much acclaimed typesetting system is almost unknown in most parts of the real world, where many a document designer has achieved professional success without ever hearing (let alone pronouncing) the word "TEX". Outside academia, the lone TEXer faces not only compatibility headaches, but also outright incomprehension from his customers, colleagues, or competitors: why would anyone want to use TEX to produce memos, two-color newsletters, full-color brochures, overhead transparencies, and other itemss — in short, anything but books that contain a lot of mathematics?

As a consultant in professional communication, I have been using TEX for all documents I have produced for my customers and for myself during the last ten years or so. Though it has turned out to be most successful, this approach is seen by most as a mere idiosyncrasy. And yet, the systematic use of my own TEX and PostScript programming gives me three unequalled advantages over using off-the-shelf software: I travel light, I can go anywhere I please, and I guarantee I'll get there.

The presentation relates the experiences of a long-time lone TEXer in the real world. It explains choices, points out advantages and limitations, and draws lessons from the experience.

2.2 The vulcan Package: A Repair Patch for LaTeX

Peter Flynn
University College Cork, Ireland
pflynn@imbolc.ucc.ie

Abstract: TeX and LaTeX systems have proved successful in providing high-quality, affordable typesetting from the desktop. This success has tended to disguise some of the less felicitous design decisions made in good faith in earlier days. These are now making it harder to keep LaTeX-based systems in line with user expectations, as the defaults remain rooted in a document model which fails to reflect reality.

Of the thousands of LaTeX packages, many have already tackled the allied problems of providing better formatting configuration management, and of adding new formatting features. This paper presents an attempt to tackle two of the more deep-seated problem: providing an improved document model and providing fixes for what many users perceive (wrongly) as bugs.

The document model is compared with current practice elsewhere in the text field; the fixes answer the top dozen or so user requests from the various FAQs and questions asked on `comp.text.tex`. The changes are presented as a standard LaTeX 2_ε package (vulcan) which allows LaTeX to match more closely the way that authors write and edit documents, as well as providing improved appearance.

2.3 New Interfaces for LaTeX Class Design Parts I and II

David Carlisle
LaTeX3 Project
david.carlisle@latex-project.org

and

Frank Mittelbach
LaTeX3 Project
frank.mittelbach@latex-project.org

and

Chris Rowley
LaTeX3 Project and Open University, UK
chris.rowley@latex-project.org

Abstract: Traditional LaTeX class files typically implement one fixed design via ad hoc, and often low-level, (La)TeX code. This style of implementation makes it much harder than is either desirable or necessary to produce classes that implement a specific visual design. Moreover, the construction of such classes typically involves a lot of work that is essentially programming and thus does not live easily with the declarative kind of design specification for a document (or range of documents) that would be produced by a professional typographic designer.

This work introduces some extensions to LaTeX that will help to provide a new, more declarative interface that can be used in class files. It is based on the idea of a *template*, which describes how to carry out some action but which provides some flexibility since its code uses the values of a set of named (keyword) parameters. The specific design for this action, as required for a particular class, is then selected by choosing values for the template's named parameters.

The plan is to provide standard *templates* for a wide range of typographic objects but, of course, new templates for new ideas can be created, possibly by adapting an existing one or by a little LaTeX programming. It is our firm belief that there will soon be a large range of templates avaliable and that it will thus be possible for the majority of class files to be implemented in a declarative way, by simply choosing suitable templates and supplying values for their named parameters.

We have working examples of the application of these ideas in most of the major areas of document design, including page layout, section headings, lists, captions.

The two talks will explain these concepts and show examples of their use covering both the current standard LaTeX designs and some more exciting new possibilities.

2.4 Writing Class Files: First Steps (workshop)

Michael Doob
University of Manitoba, Winnipeg, Canada
mdoob@cc.UManitoba.CA

Prerequisites: Some experience with LaTeX 2$_\varepsilon$ and macro writing.

Description: This workshop will cover (just) enough background to allow participants to write their own class files. The standard file `classes.dtx` will be used as a model. Topics include:

- class files and the `docstrip` concept
- the different file extensions — an alphabet of woe?
- the `classes.dtx` file
- adapting to make your own class file

2.5 Converting a LaTeX 2.09 Style to a LaTeX 2ε Class (workshop)

Anita Hoover
University of Delaware
anita@udel.edu

Prerequisites: Some experience with LaTeX 2.09 style files.

Description: This workshop will take a LaTeX 2.09 style for the University of Delaware thesis layout and show the steps necessary for converting it to a LaTeX 2ε class file:

- Do you need a package or a class?
- Can you build on an existing class?
- Incorporating your specific style
- Any special options required for your class?

3 Wednesday August 18, 1999
TeX in Publishing

3.1 Multi-Use Documents: The Role of the Publisher

Kaveh Bazargan
Focal Image Ltd, London
kaveh@focal.demon.co.uk

Abstract: Not so long ago, a typesetter was only expected to produce 'camera-ready copy' on paper or bromide. This is now rarely the case. Publishers now expect a variety of electronic files from the typesetter, both for publishing, and for archiving. The term 'typesetter' sounds amusingly outdated, but the fact that it is still being used is significant. The publishing industry, by and large, still operates in the traditional 'manuscript / edit / typeset / proofread / print' route. The process is still essentially paper-based, with the manuscript being submitted as the definitive author submission, accompanied by optional electronic files. Similarly, the typesetting is geared primarily towards the paper output, with the electronic files being supplied as an extra, after approval of CRC.

I suggest that we are in need of radical changes to these procedures so that files can be used to produce output on any medium, including paper, and I believe that these changes will benefit all concerned — authors, publishers, and typesetters.

TEX is an ideal medium to hold the definitive text in modern typesetting. Not only can it be directly edited by the author, but it can produce every type of output required directly. These include paper, PDF, SGML, HTML, XML, etc. (During the presentation, there'll be live demonstrations of this usage of TEX.) However, in order for this to work well, the document should be clean, and structured.

LATEX is probably the best standard at the moment, with a big user base. I see an important role for the publisher as that of encouraging the submission of structured LATEX files from authors. This could best be done by taking a more active role, and sending an 'author kit' to prospective authors. The class file sent out in this kit need not — and I believe, should not — be identical to that used in the final typesetting of the manuscript. The class file for the author can be designed to run on most systems without problems, e.g., use Computer Modern font. The class file used by the typesetter could be more ambitious, perhaps getting away from the standard 'TEX look', using unusual fonts, graphics packages, etc. This separation of class files frees the typesetter to be more creative in the final typesetting.

For book authors, there is another possible advantage in using structured LATEX files; namely, that the final source code can be sent back to allow the authors to work on the next edition (using the class file provided in the kit). There are very few other systems that can allow this approach. We have been using this method successfully. This could be a good selling point in encouraging authors to use TEX.

Finally, considering the advances in electronic delivery of files, I believe it is time to re-assess the standard marks used in copy-editing manuscripts. These have not changed at all to adapt to new technology. The marks are still geared towards traditional typesetting with paper output. Perhaps a new system could take into account the structure of the file, making sure that it translates well in all the output formats, including paper.

3.2 LaTeXto XML/MathML for Web-Based Publication

Frederick H. Bartlett
Springer-Verlag New York, Inc.
fredb@springer-ny.com

Abstract: At Springer-Verlag, we have been frustrated for some years now with the difficulty of putting mathematics into a web-friendly format. We have not yet found a magic bullet, but ...

The XML application MathML may be the first real tool for putting mathematics on the Web in a useful form. Suppliers of mathematical tools such as Mathematica and Maple are gearing up to use MathML as an input/output format; thus, we can look forward to a day when mathematics on the Web will be truly interactive.

It is likely that — even if MathML fulfills every bit of its promise — TeX will continue to be used for the preparation of mathematics for display and printing.

This presentation is an account of our efforts to translate author-generated LaTeX into XML. The project can be divided into four stages:

1. Normalizing (La)TeX. That is, transforming authors' idiosyncratic usages (and even more idiosyncratic macro definitions) into consistent, and consistently structured, files. The vast majority of author-generated LaTeX files can be converted easily with a minimal understanding of TeX's digestive tract; those which can't (especially plain TeX files) will require some human intervention — or increasingly sophisticated (read 'bloated') software.

2. Converting to XML. This is the easy part: changing structural LaTeX tags into XML tags.

3. Converting to MathML. And this is the hard part: It would be ideal to be able to convert LaTeX math into both presentation and content MathML coding. Unfortunately, this is, even in principle, extremely difficult. So at first we concentrate on the LaTeX-to-presentation mark-up path. Eventually, it will be possible to produce an interactive LaTeX-to-content mark-up converter for authors.

4. Going backwards. It will eventually be helpful to authors and publishers if MathML/XML can be converted back to (La)TeX, but this is not a high priority at the moment.

This talk describes something that is very much a work in progress, so a discussion period will be most welcome.

3.3 Making a Book from Contributed Papers: Print and Web Versions

Harry Payne
Space Telescope Science Institute, Maryland
payne@stsci.edu

Abstract: In my field (astronomy), it is common practice for conference proceedings to be generated from a collection of contributed papers, each of which is a stand-alone LaTeX document. Editors are expected not only to edit the individual contributions but also to combine them into a book, with a table of contents and proper page numbers, in spite of the fact that publishers generally provide little more than a style file and instructions for the individual authors.

In my talk, I will describe a set of tools that I have used on several conference proceedings projects. The basic idea is to process the individual contributions into chapters, suitable to be `\input` into a skeleton document, allowing the entire book to be processed as a single LaTeX document. All of the processing is managed by the UNIX `make` utility, and performed with `perl` and `sed` scripts. One advantage to this scheme is that references to other papers in the same volume may be supplied with a page number in a straightforward way. Another is that the entire book may then be processed with `latex2html` to produce a Web version from the same set of input files. Recent examples may be seen at

- http://www.stsci.edu/stsci/meetings/lisa3/, the third conference on Library and Information Services in Astronomy
- http://www.stsci.edu/stsci/meetings/adassVII/, the seventh annual conference of the Astronomical Data Analysis Software and Systems Conference series.

In these examples, the output from `latex2html` has itself been processed through other `perl` scripts, and a lot of hand editing. The scheme has been used for volumes in the *Publications of the Astronomical Society of the Pacific Conference Series* and the Kluwer *IAU Conference Series*. The tools will be made freely available via the Internet.

3.4 Managing Large Projects with PreTEX, a Preprocessor for TEX

Robert L. Kruse,
PreTEX Inc., Halifax, Canada
bob@pretex.com

Abstract: PreTEX is a preprocessor for TEX that supplies an author with many tools to simplify the writing and management of larger (book-length) projects.

This talk will concentrate on PreTEX's use of secondary input files and conditional typesetting in managing large projects. The user may insert location tags within a file which can then be used by PreTEX to include parts of one file within another, in any order determined by the user. Parts of a file may also be selectively typeset according to the status of various conditions. Three sample applications will demonstrate the power of these tools:

1. Consider a textbook in which answers to problems are printed separately. With PreTEX, solutions can be placed immediately adjacent to the problems, but will be printed only under the control of PreTEX commands. Hence the same input can, as desired, produce solutions with the problems, at the back of the book, or in a separate document.

2. Consider a compendium in which chapters are written by authors not in touch with one other, and each chapter is processed independently through PreTEX. Each chapter may have its own cross references, index, or contents. The editor can merge all these resources for the entire volume, supplying a bibliographic database for use by all authors, accessed by BibTEXautomatically from PreTEX.

3. Consider a large software system with computer code distributed over many files, and with documentation in the same files with the code. There may be several kinds of documentation: informal introductions, user reference material, precise specifications, programmer's comments, revision reports, and chronology. By treating each program file as a secondary file, PreTEX can combine any desired extracts of the documentation or the program code in any desired order. Identical source files can be used to construct, for example, informal user guides, user reference manuals, or complete program listings. PreTEX provides special facilities for typesetting computer programs, understanding enough of the program syntax to adjust spacing and choose special symbols.

   For program files, PreTEX provides a further utility called StripTEX, which removes the documentation and any TEX markup in the program code, yielding output that can be submitted directly to a compiler. In this way, PreTEX provides all the functions of Knuth's Web system (Weave and Tangle), but with additional capability, flexibility, and language independence.

3.5 Database Publishing with JAVA and TEX

Arthur Ogawa
TEX Consultants, California
ogawa@teleport.com

Abstract: Sun Computer intends its JAVA programming language to be the lingua franca of the World Wide Web. Sun may get their wish: dozens of books about JAVA are published every year, and there is even a conference about JAVA, JavaOne, being held in San Francisco at the same time as this TUG meeting.

If you wanted to publish a book documenting and cross-referencing all the JAVA class libraries (running to 1,000 pages and covering over 1,500 classes, organized into about 70 "packages") what would you do? If you are Patrick Chan and Rosanna Lee, you would use JAVA itself to manage the data (about 40Mb of it) and you would use TEX to format your pages.

In my talk, I will describe the criteria for selecting the formatter (i.e., TEX plus macros) for a database typesetting project, the best way of interfacing between TEX and a database engine, and some interesting (perhaps even challenging) features of the formatting work. I will also show how the success of the project enabled the author to make last-minute revisions to the book (changes necessitated by late developments in the JAVA class libraries themselves) even though this involved the reprocessing of all 40Mb of data, in less than 24 hours.

3.6 Implementing Dynamic Cross Referencing in PreTEX and PDF

Paul A. Mailhot
PreTEX Inc., Halifax, Canada
paul@pretex.com

Abstract: PreTEX is a preprocessor for TEX that supplies an author with many tools to simplify the writing and management of larger (book-length) projects. PreTEX automatically supplies some of the markup required by TEX, simplifies other commands, and provides user-friendly error diagnosis. PreTEX allows the independent processing of a project in conveniently small (chapter length) files at every stage, while integrating cross references, index and contents references, and bibliographic citations across the entire book. All of these we call Dynamic References.

This talk will concentrate on using PreTEX for on-line documents, that is, documents published on CD-ROM or the Web, for example. The popularity of on-line publication has spawned a wide variety of on-line document readers and Web browser plug-ins. These include FrameReader, Mathematica Reader, Quark Reader, and Adobe Acrobat Reader, to name only a few.

Adobe Acrobat supports cross-referencing elements in a manner somewhat similar to the method PreTEX uses for dynamic references. The PreTEX system, in fact, can automatically insert PDF (Portable Document Format) bookmarks, tags, and links for all the dynamic references in the original document. In this way, all these references (index, contents, cross references, bibliographic citations, etc.) automatically become links for navigating through the on-line document.

In addition, further PreTEX commands (ignored for the printed version) will generate page thumbnails, document contents, or icons that will launch application files directly from the PDF document. Suppose, for example, that part of a document is discussing some computer process or application. PreTEX can then place an icon in the margin (of the PDF version only, not the printed version) that will immediately launch that application or start a demonstration program. On termination, the application or demonstration will return to the same place in the document.

Dynamic referencing in PreTEX is considerably more flexible and powerful than in other systems, such as LaTeX 2_ε. Index entries can be generated in several forms; bibliographic citations can be either local to one file or global to the entire project; cross-reference labels are similar to those of LaTeX, but they can be global over all files, and the replacement text may include words, phrases, macros, and other elements, as well as counters such as section or page numbers. If desired, PreTEX can automatically perform simple edits on the replacement text. If, for example, two references used together expand to “Figure 7 and Figure 10” PreTEX can be instructed to parse this as “Figures 7 and 10”.

3.7 A Web-Based Meeting Abstract Submission System

Hu Wang
American Institute of Physics, New York
hwang@aip.org

Abstract: As one of the services provided to American Institute of Physics (AIP) member societies, we publish program books and abstract books for meetings sponsored by the societies. In this presentation, we describe a web-based abstracts submission system which, using a form template front end for users to enter their abstracts and related information in LaTeX or non-LaTeX mode, saves users' input into a database and generates LaTeX files. This is followed by a brief discussion of integrating the submission system with the publishing system.

3.8 Hyphenation on Demand

Petr Sojka
Masaryk University, Brno, Czech Republic
sojka@muni.cz

Abstract: The need to fully automate the batch typesetting process increases with the use of TEX as the engine for dynamic documents which, in turns, leads to the need for programmable hyphenation and line-breaking of the highest quality.

An overview of approaches for building *custom* hyphenation patterns is provided, along with examples. A methodology of the process is given, combining different approaches: one based on morphology and hand-made patterns, and one based on wordlists and program patgen. The method aims at modular, easily maintainable, efficient and portable hyphenation. The bag of tricks used in the process to develop custom hyphenation is described.

The presentation may be reworked into a half-day tutorial aimed at the following types of audiences:

Publishers: to help them solve their problems with hyphenation by showing how to create in-house hyphenation patterns

TEX practitioners: to examine the possibilities that the eTEX system offers in the areas of hyphenation and line-breaking

3.9 Active TEX and the DOT Input Syntax

Jonathan Fine
Independent Consultant, Cambridge, UK
fine@active-tex.demon.co.uk

Abstract: The usual category codes give TEX its familiar backslash and braces input syntax. With Active TEX, all characters are active. This gives the macro programmer complete freedom in defining the input syntax. It also provides a powerful programming environment.

The dot input syntax, like TROFF, uses a period at the start of the line as an escape character. However, its underlying element, attribute and content structure is based on SGML. It is both easy to use and easy to program for. This new syntax will be described and demonstrated.

All manner of problems connected with TEX disappear when Active TEX packages are used. For example, all input errors can be detected and corrected before they cause a TEX error message. This will make TEX accessible to many more users.

4 Thursday August 19, 1999
Fonts, Graphics, and New Developments

4.1 Alternatives to Computer Modern Mathematics

Alan Hoenig
City University of New York
ajhjj@cunyvm.cuny.edu

Abstract: It is apparently difficult to typeset mathematical documents using fonts other than Computer Modern — at least to typeset them properly. I discuss a technique for using Metafont, of all tools, to generate math fonts that are compatible with Type 1 scalable roman fonts. The idea is to alter Metafont font files so that their parameters match Times Roman, Garamond, or whatever.

The resulting meta-fonts have figures and letters which look ghastly, although the mathematical symbols are quite acceptable. Using TEX's virtual font mechanism, we construct virtual math fonts which use the math symbols and discard the ghastly letterforms in favor of beautiful ones from the Type 1 fonts.

This is a mildly technical opeation, performed automatically by my MathKit tools, freely available on CTAN. I will be discussing these tools, detailed ways in which to use them, and will be displaying many examples of math fonts created from this technique.

4.2 Convenient Labelling of Graphics, The WARMreader Way

Wendy McKay
California Institute of Technology
wgm@cds.caltech.edu

and

Ross Moore
Macquarie University, Sydney, Australia
ross@zeus.mpce.mq.edu.au

Abstract: This presentation describes a system for placing labels on included graphics in a way that does not require that the user be concerned with explicit lengths or coordinates. The full system was developed specifically for use on Macintosh computers but, due to its modularity, can be used with other systems as well.

The WARMreader (read 'Wendy And Ross', selecting either for the 'M') macro package reads symbolic information from a file, indicating the location of specially marked points to which labels are to be attached. It also provides a link to the XY-pic macros, which allow arbitrary labels to be attached to these marked points.

Of course, one also needs a method to create the files that are read by WARMreader. On the Macintosh, this is provided by a special plug-in to David Rand's Zephyr application. By simply clicking the mouse on the desired points, a file is built to contain all the required information.

4.3 Drawing Effective (and Beautiful) Graphs with TEX

Jean-luc Doumont
JL Consulting, Brussels, Belgium
JL@JLConsulting.be

Abstract: A standard approach to producing documents that include illustrations consists in typesetting text with specialized typesetting software (such as TEX) and inserting illustrations created with different, equally specialized software. To better integrate the illustrations into the typeset page, it would be nice to be able to produce or modify them directly with the typesetting software. Drawing graphs with TEX, for example, would allow one to, say, set them `\hsize` wide and `0.75\hsize` high, position labels exactly `\baselineskip` below the horizontal axis, and, especially, typeset all annotations with the same fonts, sizes, and mathematical beauty as the rest of the document.

The hybrid TEX and PostScript macros presented in this presentation take advantage of TEX's powerful approach to graph and annotate data sets in a variety of ways, in order to produce effective, beautiful, well-integrated graphs. They use TEX to draw all horizontal and vertical lines (axes, tick marks, grid lines) and set all annotations. Then PostScript is used to draw the data, as markers, lines, and areas. While fairly simple, they have been successfully harnessed to appear in a wide range of real-life applications, up to logarithmic graphs and (with some patience) complex multipanel displays. Of course, the macros are a tool for drawing final graphs, not for exploring or transforming data sets.

4.4 Viewing DVI files with Acrobat Reader: A privileged role for DVIPDF

Sergey Lesenko
Institute for High Energy Physics, Protvino (Moscow Region), Russia
lesenko@mx.ihep.su

and

Laurent Siebenmann
Université de Paris-Sud, France
lcs@topo.math.u-psud.fr

Abstract: The free DVIPDF program of the first author converts from TEX's output DVI files to Adobe's PDF (= Portable Document Format). Although DVIPDF has existed as a prototype for about three years, the uses to which it will be put by the TEX community are only gradually emerging.

This article presents one concrete application. DVIPDF has been adapted under the Windows 9X/NT operating systems to permit instant viewing of DVI files with Adobe's free Acrobat Reader.

The purpose of this mechanism is to combine the excellent size-economy and polyvalence of TEX's DVI norm with the high performance of Acrobat Reader. The immediate benefit (and the original motivation) has been to greatly extend the capacity of CD-ROMs for scientific literature. Great cost advantages are expected for those who connect to the web through a telephone modem, and for scientific research libraries on the web that have many mirror sites worldwide.

Since the 1980s, it has been traditional for PostScript printing to include such auxiliary graphics files as EPS (= Encapsulated PostScript) along with a DVI file. These should now be presented, instead, as graphics files of various "optimal" norms: JPEG or PNG for bitmaps, and PDF for vector graphics (these are not difficult to convert to EPS if needs be).

Hopefully, it will become possible to reasonably fully exploit the rich potential of Acrobat Reader with this mechanism. Hopefully, it will also soon be possible to adapt this mechanism to other platforms, such as UNIX and Macintosh.

4.5 fpTEX: A teTEX-Based Distribution for Win32

Fabrice Popineau
École Supérieure d'Électricité, Metz, France
popineau@ese-metz.fr

Abstract: This is an announcement for a new free TEX distribution for Win32. It is based on the well-known, widely used teTEX distribution for UNIX environments. The adaptation to Win32 is described: what choices have been made, how it is intended to be used on personal workstations or networks, and the specific Win32 tools that have been designed.

Special thanks to: Karl Berry, Olaf Weber, Sebastian Rahtz and Thomas Esser (and probably many others!) for their wonderful work.

Why based on teTEX? The teTEX distribution is based on Web2C, and most of the free developments around TEX are made on a Web2C basis. The teTEX distribution encapsulates Web2C into something very easy — albeit powerful — to install and use under UNIX. So naturally, I tried to replicate teTEX features under Win32. Here follows a brief description of what can be found in fpTEX, and some small differences with regard to the original version. It is interesting to note that all the source code is the very same. The difference in building the distribution lies mostly in `Makefile`s and tools. On other difference which merits special mention: the XDvi previewer provided with teTEX was not portable under Win32. However, all the core code has been reused in a new wrapper, and the result is Windvi.

The installer An InstallShield-based installer is provided. It is devised to allow personal or network installations. The user is asked for some basic configuration date. Here are some predefined schemes:

- personal installation
- network installation, Win32 environment
- network installation, Win32 and UNIX mixed environment

The installer allows a selection of packages on a per-tool basis. That means if you ask for LATEX, you will get all the LATEX packages.

The configuration tool The teTEX distribution provides a Swiss-army knife to do all the configuration management. This tool, called `texconfig`, has been adapted to the Win32 environment under the form of tabbed dialog boxes.

4.6 Managing TEX Software Development Projects

Jeffrey McArthur
Atlis Publishing Services, Maryland
jmcarth@atlis.com

Abstract: During the past few years, many articles and books available about managing software development projects have been written. Software development projects using TEX require some special attention. This presentation looks at the entire software development lifecycle as it applies to TEX and the following issues in particular:

- requirements specification
- design specification
- coding standards
- code review checklist

4.7 JAVA 2000

Timothy Murphy
Trinity College Dublin, Ireland
tim@maths.tcd.ie

Abstract: TEX and Metafont, translated into JAVA and compiled with TYA, a public-domain JIT compiler, run about 10 times more slowly than the same programs in C (without TYA, they are 20 times slower). A year ago, they ran 50 times more slowly. In a year's time, perhaps using Sun's new JIT compiler, it is reasonable to assume that the factor will be down to 2 or 3.

At that point — bearing in mind the speed with which the speed of computers is increasing — TEX-in-JAVA will be a perfectly plausible alternative to TEX-in-C; and then we shall have to weigh its lack of pace against the several advantages that JAVA has to offer, such as:

Portability: in principle, exactly the same JAVA programs should run under every OS;

Netability: Java was designed with the Internet in mind, and its adoption must allow TEX to be integrated more easily into the Web.

Modularity: Although TEX and Metafont are large monolithic programs, they are actually written in a modular style — almost as though Knuth had JAVA in mind! — and it should be simple to 'hive off' font routines, for example, as a separate `TeXFont` class, without modifying the essential code in any way. Breaking up TEX (or Metafont) in this way into a number of co-operating classes might mean that variations like pdfTEX and Metapost could be implemented as relatively small 'extensions' of one or more of these classes.

Threads: By running TEX and friends as 'threads', last-minute changes can be implemented before the thread starts; as well, the program can pause while some intermediate task is performed, before resuming where it left off.

Graphics: JAVA offers a graphical alternative to the perhaps old-fashioned text-based interface traditional to TEX.

VENDORS–PUBLISHERS

If you wish to participate as a vendor/publisher,
please register your interest by completing the Vendors-Publishers Form.

Vendor	Product	E-mail	URL
Blue Sky Research	Textures	sales@bluesky.com	http://www.bluesky.com
Cemtlomedia Corp.	TeXplus—Viewer/ Writer/Publisher	jccha@knot.kaist.ac.kr	http://www.cemtlo.com
Design Science Inc.	MathType 4	pault@mathtype.com	http://www.mathtype.com
Mir Publishers	Books about TeX in Russian	irina@mir.msk.su	
Personal TeX Inc.	PCTeX	lcarnes@pctex.com	http://www.pctex.com
TrueTeX Software	TrueTeX	kinch@truetex.com	http://www.truetex.com

ORGANIZERS

TUG '99
c/o Integre Technical Publishing
4015 Carlisle NE, Suite A
Albuquerque, NM 87107, USA
Fax: +1 505-889-8192;
E-mail: tug99@tug.org
`http://www.tug.org/tug99/`

Conference

Sue DeMeritt
Patricia Monohon

Program

Anita Hoover (co-chair)
Stephanie Hogue (co-chair)
Cheryl Ponchin

Proceedings

Christina Thiele (editor)
Anita Hoover (technical)

Courses/Vendors/Publishers

Don Deland

Publicity

Wendy McKay
Heidi Rhodes Sestrich

with the help and support of

Ross Moore :
LaTeX2html

Karl Berry :
System Administrator
`http://www.tug.org/`

Credits

Duane Bibby :
Illustrator-Cartoonist, TeX Lion Art
DBibby@aol.com

TUG '99 SUMMARY

The 20th Annual Meeting of the TEX Users Group

August 15–19, 1999

University of British Columbia · Vancouver · BC Canada

Program

SUNDAY AUGUST 15	MONDAY AUGUST 16	TUESDAY AUGUST 17	WEDNESDAY AUGUST 18	THURSDAY AUGUST 19
Registration	**Official Opening**	Jean-luc Doumont	Kaveh Bazargan	Jean-luc Doumont
Welcome Reception	Stephen A. Fulling	Peter Flynn	Frederick H. Bartlett	Wendy McKay Ross Moore
	Patrick D. F. Ion	**Vendors Presentations**	Harry Payne	Break
	Doug Lovell	Break	Break	Sergey Lesenko Laurent Siebenmann
	Break	David Carlisle Frank Mittelbach Chris Rowley	Robert L. Kruse	Alan Hoenig
	Paul R. Topping	Lunch	Art Ogawa	**Vendors Presentations**
	Workshop Eitan Gurari Sebastian Rahtz	**Workshop** Michael Doob	Paul A. Mailhot	Lunch
	Lunch	Break	Lunch	F. Popineau
	Chris Rowley	**Workshop** Anita Z. Hoover	Hu Wang	Jeffrey McArthur
	D. P. Story	**TUG Business Meeting**	Peter Sojka	Timothy Murphy
	Break		Jonathan Fine	Break
	Panel Discussion		Break	**Panel Discussion**
	Workshop D. P. Story		**Panel Discussion**	**Closing Banquet**

COURSES

Friday, August 20 & Saturday, August 21

Tentative — based on enrollment