

# Baskerville

The Annals of the UK T<sub>E</sub>X Users' Group

Editor: Editor: Sebastian Rahtz

Vol. 5 No. 2

ISSN 1354-5930

March 1995

Articles may be submitted via electronic mail to `baskerville@tex.ac.uk`, or on MSDOS-compatible discs, to Sebastian Rahtz, Elsevier Science Ltd, The Boulevard, Langford Lane, Kidlington, Oxford OX5 1GB, to whom any correspondence concerning *Baskerville* should also be addressed.

This reprint of *Baskerville* is set in Times Roman, with Computer Modern Typewriter for literal text; the source is archived on CTAN in `usergrps/uktug`.

Back issues from the previous 12 months may be ordered from UKTUG for £2 each; earlier issues are archived on CTAN in `usergrps/uktug`.

Please send UKTUG subscriptions, and book or software orders, to Peter Abbott, 1 Eymore Close, Selly Oak, Birmingham B29 4LB. Fax/telephone: 0121 476 2159. Email enquiries about UKTUG to `uktug-enquiries@tex.ac.uk`.

---

## Contents

I	Editorial .....	3
1	Portable documents meeting .....	3
2	Bits and pieces .....	3
II	Letters to the editor 1: Maths in L <sup>A</sup> T <sub>E</sub> X, Part 3 .....	3
III	Letters to the editor 2: Maths in L <sup>A</sup> T <sub>E</sub> X, Part 3 .....	5
IV	Letters to the editor 3: DANTE .....	6
V	Portable Documents: Why Use SGML? .....	7
1	Introduction .....	7
2	Documents .....	7
3	Portability .....	7
4	Who needs portable documents, and why? .....	7
5	Examples of successful portability .....	8
6	Achieving portability .....	8
7	Why use SGML? .....	8
7.1	An example .....	8
8	The future .....	8
VI	Formatting SGML Manuscripts .....	9
VII	HTML & T <sub>E</sub> X: Making them sweat .....	14
1	Markup .....	14
2	The World Wide Web .....	14
2.1	HTML Markup .....	15
2.2	Implementation .....	15
2.3	Presentation .....	17
3	Publishing with HTML .....	17
3.1	Printing from HTML .....	18
3.2	Problems .....	18
4	Development .....	18
VIII	The Inside Story of Life at Wiley with SGML, L <sup>A</sup> T <sub>E</sub> X and Acrobat .....	20
1	Introduction .....	20
2	Setting the Scene .....	20
3	Complementary not Competitive .....	20
4	SGML in Practice .....	20
4.1	Production Considerations .....	21

4.2	Problems Encountered	21
5	$\LaTeX$ in Practice	21
5.1	Steps in the Process	21
6	Acrobat at Arm's Length	22
7	Conclusion	23
IX	Theory into Practice: working with SGML, PDF and $\LaTeX$ at Elsevier Science	24
1	The Company	24
2	The move into electronic publishing	24
3	The concept of Computer Aided Publishing (CAP)	24
4	Practicalities: How we do it	24
5	$\TeX$ and $\LaTeX$	25
6	Practical Problems	25
7	The Future	26
X	SGML and $\LaTeX$	27
0.1	Further reading	29
XI	Questions and Answers	30
XII	Maths in $\LaTeX$ : Part 4, Numbered and Unnumbered Things	33
5	Numbered and Unnumbered Displays	34
5.1	Unnumbered Maths displays	34
5.2	Unnumbered word displays	34
5.3	Numbered Maths displays	34
5.4	Numbered word displays	34
5.5	Numbering equations within sections	35
5.6	One-off numbering of equations	35
5.7	Subsequences of equations	36
6	Theorems and their friends	36
6.1	Basics	36
6.2	Named theorems	37
6.3	Sequences of numbering	37
6.4	Unnumbered environments	38
6.5	Other systems of numbering	38
6.6	Changing the fonts	38
6.7	Proofs	39
6.8	Questions and Exercises	39
7	Other numbered things	40
7.1	Numbered lists	40
7.2	Footnotes	40
XIII	One by one the guests arrive	41
1	Why,	41
2	Conclusion	42
XIV	Something is happening, but you don't know what it is	43
XV	Malcolm's Gleanings	45
1	Spivaking anyone?	45
2	Stability or stasis	45
3	<i>TUGboat</i>	45
4	A few last words	45

---

## I Editorial

---

### 1 Portable documents meeting

Most of the articles in this issue are based on talks given at the UKTUG's successful *Portable Documents* meeting on January 19th at the Bridewell Theatre, Fleet Street. We are very grateful to Jonathan Fine for his hard work in transcribing the panel session, and to him, Malcolm Clark and Carol Hewlett for working so hard to make the day a success. Jonathan also provided the following summary of the papers:

*Why Use SGML?* by David Barron gives an overview of why portable documents are important, and the value of SGML. *Formatting SGML Manuscripts* by Jonathan Fine first describes some typographic problems, and then describes a new TeX macro package which will typeset directly from SGML document instances. *HTML & TeX* by Peter Flynn gives an overview, from the HTML document side, of the rapidly expanding World Wide Web, and expresses opinion as to its future. *The Inside Story of Life at Wiley* by Geeti Granger is just that. It provides a valuable insight into the real-world problems and solutions that arise in a busy production department. *Theory into Practice ... at Elsevier Science* by Martin Key describes the progress they have made towards holding all new journal articles in standard electronic form. *Questions and Answers*, prepared for publication by Jonathan Fine, is an edited version of the lively and informative Panel Discussion which closed the Bridewell meeting. *SGML and L<sup>A</sup>T<sub>E</sub>X* by Horst Szillat compares the two, and argues that there is a need to manipulate the data during the conversion process. (This paper was not presented at Bridewell.)

### 2 Bits and pieces

Readers will already have noticed that this issue of *Baskerville* has been expanded to make room for all the Bridewell papers, as well as the usual more traditional TeX fare. Thankfully, almost no room is left for editorial comment. But we do need room to pass on a correction from Arthur Smith about the HyperTeX project described in *Baskerville* 4.5; the mailing lists have changed:

I am currently maintaining two mailing lists based at `snorri.chem.washington.edu`: `hypertext-announce` is for major announcements of new software or macros, and `hypetex-dev` is for detailed discussions of the development of HyperTeX. Send e-mail to `majordomo@snorri.chem.washington.edu` with subscription and information requests, or contact me directly (`asmith@mammoth.chem.washington.edu`).

and to pass on important news from Peter Abbott. Firstly, *Thinking in Postscript* by Glenn Reid is out of print and Addison-Wesley say unlikely to be reprinted. So don't order it. Secondly, Peter has negotiated group licences for the shareware products OzTeX version 1.8; DVIPS, METAFONT and the Alpha text editor for Macintosh; and the Eddi4TeX TeX shell and editor for DOS. Contact Peter for details if you want to take advantage of these licences; they are only available to individual UKTUG members.

## II Letters to the editor 1: Maths in L<sup>A</sup>T<sub>E</sub>X, Part 3

I am delighted to see the esteemed Professor Bailey so wholeheartedly deprecate the re-definition of a Plain TeX command by an adjunct package (*AMSTeX*); what is less clear is why she does not vent the same spleen on the author(s) of L<sup>A</sup>T<sub>E</sub>X, who have pre-empted many more fundamental commands than just `\emptyset`.

Yours etc,

Philip Taylor

*The author replies:* I think that there is some confusion here, as well as a difference of opinion. I was complaining about a redefinition in the file `amstex.sty`, which is a package explicitly for use with the L<sup>A</sup>T<sub>E</sub>X format. That is a different matter from a redefinition in a format file, such as *AMSTeX*.

In my opinion there is a world of difference between changing the definition of the name of a glyph, which any

author may want to use, and changing the definition of a programming command. The former should absolutely not happen, because it affects people who have no idea how to get round it. I am agnostic about the latter, but would not be surprised if it were necessary in a format file.

R. A. Bailey, QMW

---

### III Letters to the editor 2: Maths in L<sup>A</sup>T<sub>E</sub>X, Part 3

David Carlisle

---

The comments on the inadvisability of redefining user level commands are valid, but the example in question, `\emptyset` in the AMS packages, is just due to an error in the first printing of *The L<sup>A</sup>T<sub>E</sub>X Companion* (as noted in `compan.err` in the L<sup>A</sup>T<sub>E</sub>X distribution.) The ‘amssymb’ package does not redefine `\emptyset`. It still looks like a 0 with a line through it. The same glyph as in plain T<sub>E</sub>X. `\varnothing` is a slashed-circle.

Actually this raises an interesting side issue. The error in the Companion printing was due to an error in the styles for *Lucida* fonts. (The Companion does not use the cm or AMS fonts). As *Lucida* does provide both glyphs, it was simply an error to have the names interchanged, but consider a hypothetical situation of a math font family that only provides one slashed-closed-curve. How visually dis-similar to  $\emptyset$  may it be before it becomes unacceptable to assign it to the command `\emptyset`? For text fonts large differences are acceptable. ‘Q’ does not look much like ‘q’ but both are accessed by ‘Q’ and any differences are accepted by the reader as differences in font design. In mathematics the situation is not at all clear...

---

#### IV Letters to the editor 3: DANTE

Philip Taylor  
RHBNC;  
Technical Director, NTS project

---

At the end of the first paragraph of “Malcolm’s Gleanings”, the author urges readers to ‘Expunge . . . from your minds [the malicious and/or mischievous gossip that “DANTE was reluctant to support Haralambous’ and Plaiçe’s Omega project since it was thought to compete with NTS”]’. Despite having a close and mutually beneficial relationship with DANTE, such allegations were completely new to me, as I suspect they were to the majority if not all of your readers: one must therefore ask at whose door these allegations of malice or mischief should properly be laid?

I would further add that I know of no-one, either within DANTE or within the NTS team, who would wish John and Yannis anything less than complete success with their project; the world will be enriched, not diminished, by the availability of alternative derivatives of T<sub>E</sub>X.

Yours etc.

---

## V Portable Documents: Why Use SGML?

David Barron  
Department of Electronics and Computer Science  
University of Southampton

---

### 1 Introduction

In this article we present a few ideas as a framework for the discussion of portable documents. We address a number of questions:

- What are portable documents?
- Who needs them, and why?
- How to produce them, now and in the future

### 2 Documents

Traditionally, a document was a file (or a deck of cards), and consisted solely of text. Today, documents are typically *compound*, a mixture of text and graphics (bit-map or line art) that can be rendered on paper or screen. Additionally, they may include hypertext links (in which case they can only be viewed on screen). A recent development is the ability to incorporate video and sound in a compound document, either embedded within the document or linked by a pointer: such a document is a *multimedia* document. Hypertext-style links may also be included to form a *hypermedia* document: evidently, multimedia and hypermedia documents can only be 'read' on a suitably equipped computer system.

World Wide Web (WWW) documents are a special case of compound hypermedia documents where the links are to other documents elsewhere on the Internet. They can be regarded as virtual documents, in the sense that the whole document never exists as a single identifiable object. More generally, we can define a *virtual document* as a structured collection of information from which instances of documents and other resources can be derived. Examples include:

- The Oxford English Dictionary which exists as a database from which are derived various printed editions (Shorter, Concise, Pocket etc.), as well as the CD-ROM version
- Critical editions of a literary text, where a single source 'document' contains all the variations, and can be printed out using different variants as the base text

### 3 Portability

The definition of portability that we shall use in this discussion is the ability to transmit the document digitally (over a network, or on a disk or CD-ROM) and re-create a faithful rendering of the document after transmission, if need be on a different hardware and/or software platform from that on which the document was originally created. It is important to observe that there are three different forms in which the text and graphics in a document might be re-created:

- with absolute visual fidelity
- with approximate visual fidelity
- retaining content only

### 4 Who needs portable documents, and why?

Three different needs for portable documents can be adduced

1. Publishers need them in order to distribute electronic books and journals
2. Communities with common interests who need to share information need them. An example is a scientific research community whose members use diverse hardware and software
3. Librarians responsible for digital archives need portable documents, since they cannot assume that a particular hardware/software platform will exist in perpetuity

## 5 Examples of successful portability

- Computer science researchers and software manufacturers distribute documents as PostScript files. This works well if the fonts employed are restricted to the basic 35, and the use of Adobe Acrobat (pdf files) increases portability when other fonts are used.
- The Physics pre-print library at Los Alamos National Laboratory is used by many physicists world-wide: over 10,000 retrievals per day are reported. The archive holds pre-prints in  $\LaTeX$  and PostScript formats (figures in PostScript only). This is successful because the Physics community has for some years used  $\TeX$  as its preferred means of exchanging information.
- WWW documents are highly portable, since their rendering is (almost entirely) determined by the browser software, and the use of a common mark-up language (HTML) ensures portability

## 6 Achieving portability

At first sight it appears that portability might be achieved by agreeing standards (e.g.  $\LaTeX$ , PostScript, ODA, HTML). At present there is too much choice, and no obvious winner, especially in hypermedia documents. This is a sign of an immature technology. Another important fact to take into account is that it is difficult to impose standards in some environments (e.g., academia) where personal preferences lead to the equivalent of religious wars.

Particular problems in achieving portability arise from varying fonts and character codes e.g. in handling European languages. Unicode will go a long way towards solving the character codes problem.

## 7 Why use SGML?

SGML provides a formal and portable definition of document structure. SGML syntax can define a hierarchical structure of embedded document parts, and can associate a type with each component in the hierarchy. By associating a rendering definition with each type of component, it is possible to achieve a portable document. In particular, SGML provides a uniform archive format for a library of portable documents.

### 7.1 An example

Suppose it is required to maintain a library of technical documents in an environment where some authors use  $\LaTeX$ , whilst others use Microsoft Word. We can define an SGML DTD for the document structure, together with  $\LaTeX$  and Word styles to define the rendering. This opens up three possibilities:

1. Author in SGML and use a tool to produce a  $\LaTeX$  or Word version from which the printed version can be produced
2. Author in  $\LaTeX$  and use a tool to translate to SGML to produce the archive copy
3. Author in Word and use a tool to translate the RTF form to SGML to produce the archive copy.

In addition to the SGML version of the documents, the archive must contain the Word and  $\LaTeX$  style files and the translation tools. Once this is done, anyone can collect a document, the required style files and tools and produce a copy of the document. This will of course only work for text documents. For any document with graphics content, and for hypermedia documents, more is required. This is possible in principle, but much remains to be done

## 8 The future

A combination of SGML and OpenDoc is probably the best way forward. OpenDoc provides an architecture for portable documents: it treats a document as a container for a collection of ‘parts’, each of which can have other parts embedded within it. Each type of part has associated programs to edit and render it, so that documents can be re-created with varying degrees of fidelity depending on the availability of rendering software for the particular varieties of parts that it includes.

OpenDoc is a dynamic architecture, and assumes that a new type of part may occur at any time. In principle SGML can be used to describe the static structure of an OpenDoc document, providing the final link in the portability chain.

---

## VI Formatting SGML Manuscripts

Jonathan Fine

---

This article is about typography, SGML, T<sub>E</sub>X, and SIMSIM, which is a new T<sub>E</sub>X macro package. Close by are copies of several of the OHP transparencies. They were typeset *directly from an SGML document instance* using SIMSIM.

First some words about the title slide. Documents can be formatted for several purposes. They may be typeset for printing, or for conversion to Adobe PDF format. They might be formatted for viewing on a computer monitor, as is done by the WEB browsers for HTML. They might be formatted for display and alteration by a visual or WYSIWYG editor. Formatting is the process of supplying fonts, dimensions, line and page breaking rules and so forth, so as to produce a representation of the document that is (we hope) well adapted to the display medium and the needs of the user. Rendering will convert this formatted document into bitmaps or whatever that can be displayed or printed.

In my opinion SGML is as important for structured documents as ASCII is for character sets (and SQL is for databases). It is the standard that will allow different machines and different software programs to share documents. In the title I use the words ‘manuscripts’ to emphasise that my focus is on human communication from author to reader, and not transference of bytes from one machine to another. Human beings have special qualities, which can be reflected in the manuscripts they produce. More on this later.

Still on the title slide, the subtitle ‘Much Ado about Nothing’ has two meanings. The first is that in five to ten years the formatting of SGML manuscripts will be no big deal, just as today Postscript is nothing very special. The second is that success requires taking pains or ‘making much ado’ over the spaces. Which brings us on to the second slide.

### *Spaces Between Words*

Typography is not the only art where a sound sense of space is vital. Architecture and music are others. The quotation from Schnabel expresses my view beautifully. It is one thing to get the fonts and sizes right (to play the notes on the score) and another to get the little pauses or spaces right, and also the timing of the line and page breaks. In *The T<sub>E</sub>Xbook*, Knuth quote Jan Tschichold “Every shape exists only because of the space around it. . . . Hence there is a ‘right’ position for every shape in every situation. If we succeed in finding that position, we have done our job.” Much of the typographic art involves getting the space right. Getting the choice of fonts right is another skill.

Even if we cannot reach the subtle virtues just expressed, we should strive to avoid gross errors. I’m sure we have all seen two words on a page with an extra space between them, as compared to their neighbors. Often this happens because the author has for some reason placed two spaces between the words (this is the sort of things that humans are good at doing) both of which have been treated as significant by the subsequent processing. T<sub>E</sub>X’s default reading rules automatically solve most of these problems, but not when braces for emphasised text and the like are present.

The writing of this article (in L<sup>A</sup>T<sub>E</sub>X) provided an example of this. In an earlier version I had written

```
\subsection*{ Who owns what?}
```

and the like to begin subsections. This results in an unwanted space at the start of the title. Like so:

### *Who owns what?*

The making of books involves lots of co-operation, and the participants benefit when there are clear boundaries and responsibilities. For example, many authors expect their spelling and punctuation to be corrected during the publishing process, but object to their words being otherwise changed. Newspaper journalism necessarily has different rules, as does academic journal publishing. But as a general rule the author supplies the words, the formatter the spaces. Problems arise if the author has control over spacing, or fonts for that matter. During production copy-editing and other changes will be made to the author’s words. If supplied as a computer file, the author can reasonably expect to be sent back another computer file just like the one that was sent in, but containing the words as actually printed. This returned file should not exercise any control over the spaces between words, for neither did the author’s original file.

Punctuation is a great problem. By and large, the author should supply the correct punctuation mark or logical structure. The formatter must choose the font and the spacing around the punctuation mark. This will depend on the rules of style required by the publisher. So at least three parties are involved. Should the design or rules or style used by the formatter be changed, so too may the punctuation marks used. The more that can be programmed into the software, the less need there is for human action. There will always be exceptions. Production staff will need on occasion to impose their will on the software’s production of the formatted document.

*reprinted from Baskerville*

*Volume 5, Number 2*

<p style="text-align: center;"><b>FORMATTING SGML MANUSCRIPTS</b></p> <p style="text-align: center;">- or -</p> <p style="text-align: center;"><b>MUCH ADO ABOUT NOTHING</b></p> <p>UKTUG and BCS-EPSP meeting</p> <p>(c) Copyright 1995 Jonathan Fine 203 Coldhams Lane Cambridge CB1 3HY</p>	<p style="text-align: center;">SPACES BETWEEN WORDS</p> <p style="text-align: center;"><i>The notes I handle no better than many pianists. But the pauses between the notes — ah, that is where the art resides.</i></p> <p style="text-align: center;">Artur Schnabel (1885–1951)</p> <p>Basic to quality are the spaces between words, the breaking of text into lines, and the breaking of lines into pages.</p> <p>Another basic is the space separating vertically stacked elements.</p> <p>Designers understand such things.</p> <p>When the spacing between words or elements is wrong, there is no remedy to make the result good.</p> <p style="text-align: right;">1</p>	<p style="text-align: center;">WHO OWNS WHAT?</p> <p>The words belong to the author.</p> <p>The spaces between the words belong to the formatter (which should be person and program working in harmony).</p> <p>Each participant needs to respect the others.</p> <p>Punctuation is a battlefield. It is neither word nor space, but shares qualities with both. Consider:</p> <ul style="list-style-type: none"> <li>▪ Rules of style.</li> <li>▪ Quote marks versus quote font.</li> <li>▪ Depends on language.</li> <li>▪ Interaction between space, punctuation and change of font.</li> </ul> <p style="text-align: right;">2</p>
<p><b>MANUSCRIPT PROBLEMS</b></p> <p>Authors are not yet always perfect. Just because it parse without error, that doesn't mean it is without error.</p> <p>Should the messages</p> <pre>&lt;mess&gt;Hello world!&lt;/&gt; &lt;mess&gt; Hello world! &lt;/&gt; &lt;mess&gt; Hello world ! &lt;/&gt;</pre> <p>be formatted identically? And how should</p> <pre>one &lt;bold&gt; two &lt;/&gt; buckle &lt;bold&gt;my&lt;/&gt; shoe three&lt;bold&gt;four&lt;/&gt; close&lt;bold&gt; the &lt;/&gt;door</pre> <p>be formatted? Is it the author, parser or formatter who fixes such problems?</p> <p style="text-align: right;">3</p>	<p><b>THE FLAVOUR OF SIMSIM</b></p> <p>The SGML declarations</p> <pre>&lt;!ELEMENT par ANY&gt; &lt;!ATTLIST par     font (rm bf it) rm &gt;</pre> <p>together with the code</p> <pre>def (par) // links to &lt;par&gt; {     paragraph     {         // parameters go here     }     (par font) // attribute } def (par*rm) // name token // ... etc</pre> <p>tell SIMSIM what to do.</p> <p style="text-align: right;">6</p>	<p><b>FIVE IMPORTANT QUESTIONS</b></p> <p><i>When can I get SIMSIM?</i> I'm working on it! Hope for the first test release within months. This depends on clients' non-SGML requirements. Any takers?</p> <p><i>Can SIMSIM do tables and math?</i> Yes. SIMSIM can be made to do anything T<sub>E</sub>X can do.</p> <p><i>Does SIMSIM implement all of SGML?</i> No. For markup minimization, validation etc., use with a parser.</p> <p><i>Is SIMSIM compatible with L<sup>A</sup>T<sub>E</sub>X?</i> Is L<sup>A</sup>T<sub>E</sub>X compatible with SGML?</p> <p><i>What will it cost?</i> SIMSIM has been five years in the making</p> <p style="text-align: right;">7</p>

### Manuscript Problems

These will, in an ideal world, never arise. In an ideal world others do all they can to prevent or solve your problems. And we do all we can to help others. In reality the author might be preparing the manuscript using an ordinary text editor, or a word-processor with an SGML add-on. There are likely to be stray spaces and carriage returns scattered across the file. There might even be space between the last word of a sentence and the closing period or other punctuation! Particularly if an end-tag intervenes. If not ignored, if they influence the final printed page, then a few authors will discover and use this feature. Others will be distracted from the writing of words by the need to get the spaces 'right'. But we have agreed that the spaces belong to the formatter. Thus, the three 'Hello world' messages should be formatted identically. To do otherwise is to allow the author power over spacing.

For most elements it is reasonable to assume that their boundaries do not divide words. And also that between words a space should be supplied. Thus, each line in the displayed nursery rhyme should be formatted in the same way. The formatter should ignore 'extra' spaces and supply those that are 'missing'. More subtle is this. What is the natural size of space to provide between a bold word and a word in the default (say roman) font? This is a typographic question,

and so has nothing to do with which element (if any) the space character appears in. Should it be a bold-sized space, a roman-sized space, the larger, some average, or some other value.

(The OHPs have been set, for simplicity, with a space between characters depending only on current font size, but not font style. Where speed is more valuable than typography, as when an author is writing the words of a manuscript, or when the display device is a computer monitor incapable of subtle expression, this is the right choice. A quality publisher might wish to specify more closely the interword spacing.)

The thrust of this slide is that the formatting process cannot assume that the input file is ‘just so’ and correct for the intended processing. More likely it is an electronic manuscript, with electronic analogues to the physical imperfections that paper manuscripts present. We do hope, however, that it can be read.

*What is T<sub>E</sub>X the program?*

T<sub>E</sub>X is the portable program *par excellence*. It also has very few bugs. It is stable across time. It has an ethos different from commercial software, which often charges maintenance for bug reports to be responded to. With T<sub>E</sub>X one is given a modest monetary reward for finding a bug.

It is worth remembering that L<sup>A</sup>T<sub>E</sub>X is not only a macro package but also an input file syntax. Because T<sub>E</sub>X is programmable, no fixed input syntax is required. Given sufficiently tricky macros, the mighty lion that is T<sub>E</sub>X can be made to imitate other beasts, such as the unforgetting elephant that is SGML. SIMSIM is just such a set of macros.

(The usual T<sub>E</sub>X approach, when confronted with SGML files to typeset, is to translate into L<sup>A</sup>T<sub>E</sub>X or the like before calling on T<sub>E</sub>X to do the typesetting. However, it seems to me that this approach cannot but fail to give the author control over spacing, and to mishandle manuscript problems, unless the translation process is extremely sophisticated. It will need to know about the typography intended for each element and also the character data attributes. Add to this the legendary problems L<sup>A</sup>T<sub>E</sub>X has with verbatim in titles and so forth, and the limitations should become apparent. Translation to L<sup>A</sup>T<sub>E</sub>X might have been the best there was available, but it is certainly not the best that is possible.)

*What is SIMSIM?*

This brings us to the final part of the talk, which is a software announcement. The OHPs were typeset using a preliminary version of a T<sub>E</sub>X macro package SIMSIM that I have been developing for several years, and which is close to completion. The English word ‘sesame’ is already a registered computer software trademark, so I have chosen to use the Arabic word ‘simsim’. Both are descended from an Akkadian word, current in Mesopotamia at least 4500 years ago. Simsim is one of the oldest words known to humanity. It is also the key in the classic story of Ali Babar.

There are two sides to SIMSIM. Input and output. Input is SGML and also style files. Output is pages formatted by T<sub>E</sub>X. The title slide of the talk was typeset from:

```
<title-page title =  
"FORMATTING / &SGML / MANUSCRIPTS /  
- or - /  
MUCH ADO / ABOUT /NOTHING"  
>  
  
<par> UKTUG and BCS-EPSC meeting </>  
  
<ol>  
<li> (c) Copyright 1995 </>  
<li> Jonathan Fine </>  
<li> 203 Coldhams Lane </>  
<li> Cambridge </>  
<li> CB1 3HY </>  
</ol>  
  
</title-page>
```

Notice that the title has been entered as an attribute value, with the line breaks denoted by forward slash ‘/’ or solidus characters. This is a notation in wide use for displaying line breaks in verse quoted as flowing text within a paragraph. Suppose one were presented with the title slide and were asked to encode as an SGML element. This is the sort of thing that the Text Encoding Initiative Guidelines were developed for. One would record that it was a title page, that such and such was the title text, and so forth. It is this approach that led me to use the solidus to denote line breaks in the title text. This then is the sort of input manuscript that SIMSIM will be dealing with. Note that the

formatter has not been misled by the irregular spaces in the title attribute value. The `&SGML` is an entity reference. In the title it produces itself in the current font, but elsewhere it is appearing in a smaller font. This is done using  $\TeX$ 's macro capabilities.

### *The Flavour of SIMSIM*

The parsing of an SGML manuscript makes the data within it available to the formatting (or whatever) application. There is even a specification (the Element Structure Information Set) of what data is available and when. Built into SIMSIM is an SGML parser. Writing a SIMSIM style file is a matter of linking  $\TeX$  actions to SGML events, such as the parsing of a start tag. The less technically minded might like to skim the following description as to how this is done.

Another part of SIMSIM is an enhanced programming environment for the writing of  $\TeX$  macros and SIMSIM style files. Within a SIMSIM macro file the characters `(par)` denote a token that is called at the end of the parsing of a `<par>` start tag. It is up to the application or style file to define this token to perform the required actions.

Start tags can carry attributes. The characters

```
(title-page|title)
```

in a SIMSIM file represent a control sequence whose expansion is the text read by the parser as the value of the (character data) attribute `title` of the `title-page` tag. It is then up to the style file to typeset this data, or to write it to a file, or to otherwise dispose of it.

The other main type of attribute is the name-group. Loosely, this corresponds to the 'radio buttons' that graphical user interfaces provided. Each such attribute has a short finite list of possible values. For example, the HTML `IMG` tag has an `ALIGN` name group attribute, whose values can be `top`, `middle`, or `bottom`. Because SIMSIM incorporates an SGML parser, the style file need not worry about getting this information. Indeed, great errors are liable to occur if it attempts to do so. Rather, the parser makes this data available for the application to use.

For example, with the HTML `ALIGN` name group attribute the process goes like this. Within the SIMSIM programming environment the characters

```
(img|align)
```

represent a token whose expansion will be set by the parser to be one of

```
(img*top)
```

```
(img*middle)
```

```
(img*bottom)
```

according to the option selected by the author of the manuscript. The style file should assign appropriate values to the three tokens above, for example

```
let (img*top)      = vtop
```

```
let (img*middle) = vbox
```

```
let (img*bottom) = vcenter
```

(these are illustrative values, and are not necessarily sensible) and then

```
(img|align)
```

```
{
  // the image goes here
  ... ..
}
```

will cause the image to be processed in accordance with the attribute value specified in the manuscript. This is all rather easier to do than to explain. Similar mechanisms are provided to link actions to `SDATA` entities.

The observant reader may notice that I have played fast and loose with the case of tag and attribute names. For the reference concrete syntax (used by almost all SGML applications) these names are to be converted to uppercase when read. (This is controlled by a parameter in the SGML declaration.) This is in practice quite important, and so SIMSIM converts to uppercase when it parses tag and attribute names, and the same with the programming environment.

### *Five Important Questions*

This slide is my attempt to anticipate the questions the audience would like to ask. (The untechnical should stop skimming.) To amplify my answers, I am looking for SGML-aware  $\TeX$  users who would like to be early users of SIMSIM. Tables and math capabilities will, I hope, be developed to meet customers' specific needs. I do not think it best that I try to anticipate their requirements. So much will depend on the SGML DTDs they use, or intend to use. Please contact me if you have any specific questions, and particularly if you are interested in being a test site.

At the meeting I was asked some good questions. Firstly, it is possible to have the processing attached to a tag

depend on the context? The answer is yes. For example, the bulleted items on slide two are `<li>` elements, as on the title page, but within a `<ol>` rather than `<ol>` list. This is because the action attached to a tag is held as a T<sub>E</sub>X control sequence token, whose meaning can be changed just like any other control sequence. So the token represented by `(li)` can change the meaning attached to `(li)`. (In fact this may not be the best method, there are other ways.)

Another question was how does it relate to L<sup>A</sup>T<sub>E</sub>X? So far as I am concerned there is no relation with L<sup>A</sup>T<sub>E</sub>X, and no means of converting documents from one form to another. Or style files for that matter. SIMSIM and L<sup>A</sup>T<sub>E</sub>X both start with uninitialised T<sub>E</sub>X, but from there proceed in different directions and with different assumptions. I don't see any interaction between the SIMSIM and the L<sup>A</sup>T<sub>E</sub>X worlds, and if somebody creates one, that's not my doing. A related question (motivated by legacy documents perhaps) is whether, if you have well structured T<sub>E</sub>X documents, you can get something like SGML out of it. My answer is that probably you can, but that is not the problem I set myself, and not a problem I have plans to solve.

Performance was another question. How long would it take to process a long document? This depends on the computer one has, and on the mix of text and markup in the document. Preliminary tests indicate the same order of speed as L<sup>A</sup>T<sub>E</sub>X. And do I have a manual? At the moment it's not developed to such a point that I can offer manuals. But I'd like to. I want it to be a proper product. At this point it is in the process of development and I'm looking for clients who'd like to take some risk with me, or at least make some effort. I also want to supply support. Further to that, I was asked, will I be offering maintenance costs (the usual commercial practice) or rewards (Knuth's practice with T<sub>E</sub>X)? After the laughter had died down, I declined to answer the question, explaining that I did need to earn money. This was the last question.

```
#!/bin/sh

echo Content-type: text/html
echo

cat <<EOH
<html><head><title>Date and time</title></head><body><p>It is now
EOH
date
cat <<EOT
</p></body></html>
EOT
```

**Figure 1.** Example of a Unix shell script to return the date and time as a HTML file

---

## VII HTML & T<sub>E</sub>X: Making them sweat

Peter Flynn  
University College, Cork

---

### *Summary*

HTML is often criticised for its presentation-oriented conception. But it does contain sufficient structural information for many everyday purposes and this has led to its development into a more stable form. Future platforms for the World Wide Web may support other applications of SGML, and the present climate of popularity of the Web is a suitable opportunity for consolidation of the more stable features. T<sub>E</sub>X is pre-eminently stable and provides an ideal companion for the process of translating HTML into print.

### 1 Markup

HTML, a HyperText Markup Language[1], is the language used to structure text files for use in the World Wide Web, an Internet-based hypertext and multimedia distributed information system. HTML is an application of SGML, the Standard Generalized Markup Language, ISO 8879[3]. Contrary to popular belief, neither SGML nor HTML is new: SGML gained International Standard status in 1986 and HTML has been in use since 1989.

SGML is a specification for writing descriptions of text structure. In itself SGML does not *do* anything, any more than, say, Kernighan and Ritchie's specification of the C language[4] *does* anything: users and implementors have to do something *with* it. It has been slow to achieve popularity, partly because writing effective Document Type Descriptions (DTDs) is a non-trivial task, and partly because software to make full use of its facilities has traditionally been expensive. It was therefore seen as a 'big business only' solution to text-handling problems until the popularisation of HTML owing to increased use of the World Wide Web. Since 1992 the software position has also improved considerably — an extensive list of tools is maintained by Steve Pfeffer at UIO[6].

### 2 The World Wide Web

WWW (W3 or just 'the Web') is a client-server application on the Internet. Users' clients ('browsers') request files from servers run by information providers and display them, using the HTML markup embedded in the text to render the formatting. Some of the markup can provide filenames for the retrieval of graphics as illustrations, or act as anchor-points for links to other documents, which can be further text, or graphics, sound or motion video. This latter capability gives the Web a hypertext and multimedia dimension, and allows crosslinking of files almost anywhere on the Internet.

Because the HTML files are plain text with embedded plain text markup, in traditional SGML manner, they are immediately portable between arbitrary makes and models of computer or operating system, making the Web one of the first genuinely portable, multiplatform applications of its kind.

```

<html>
  <head>
    <title>Fleet Street Eats</title>
  </head>
  <body>
    <h1>Where to eat in Fleet Street</h1>
    <p>There are many restaurants in the City, from
      fast-food joints to <i>haute cuisine</i>.</p>
    ...

```

---

**Document title:** Fleet Street Eats

## Where to eat in Fleet Street

There are many restaurants in the City, from fast-food joints to *haute cuisine*.

...

**Figure 2.** Example of HTML markup and possible rendering

### 2.1 HTML Markup

An example of simple markup and an appropriate rendering is illustrated in Figure 2. The conventions of SGML's Reference Concrete Syntax[3] are used, so markup 'tags' are enclosed in angle brackets (less-than and greater-than signs), in pairs surrounding the text to which they refer, with the end-tag being preceded by a slash or solidus immediately after its opening angle bracket.

The rendering is left almost entirely to the user's client program, as there are almost no facilities within HTML for the expression of appearance apart from a minimal indication of font change (italics, boldface and typewriter-type). Indeed, most recent browsers allow the *user* arbitrary control over which fonts, sizes and colours should be used to instantiate the tagged elements of text.

### 2.2 Implementation

HTML was devised for the Web by non-SGML-experts who saw it as an ideal mechanism for implementing plain-text portability while preserving sufficient structural information for online rendering: one of the classical reasons for adopting SGML. It is now becoming standardised by an IETF working group who have produced a draft specification in the form of a formal DTD[1]. Because of the need to allow this specification to model existing 'legacy' documents (most of which would be regarded as fragments rather than document instances), as well as provide for more robust usage, the current DTD has two modes: a non-rigorous 'deprecated' mode for describing the legacy and a 'recommended' mode for creating and maintaining files in conventional form.

HTML is sufficient for minimal documents, providing the structural and visual features shown in Figure 3. A future version (3.0) is being developed by the IETF Working Group, which will allow the description of mathematics, tables and some additional visual- and content-oriented features.

Despite the coming improvements, HTML is likely to be joined in the Web by other DTDs in future. One well-known SGML software house already has a prototype browser which can handle instances of arbitrary DTDs, given sufficient formatting information. This would make it possible to use the Web for transmission and display of documents using other SGML applications such as CALS (US Military), DocBook (O'Reilly/Davenport), the TEI (Text Encoding Initiative) and corporation-specific DTDs (such as those of Elsevier).

The next version of the DTD, HTML3, contains specifications for mathematics, tables and some additional elements for content-descriptive material, as well as a few extra visual keys such as an ALIGN attribute for positional specification. Most of this work is being implemented on a test basis in the Arena browser (Unix/X only at the moment) at CERN.

Although Web browsers can reference files by any of several methods (HTTP, the Web's 'native' protocol; FTP; Telnet; Gopher; WAIS; and others) by using the URL (Universal Resource Locator: a form of file address on the Internet), the most powerful tool lies at the server end: the ability of servers to execute scripts, provided their output is HTML. A trivial example is shown in Figure 1, which returns the date and time.

Such a script can contain arbitrary processing, including the invocation of command-line programs and the passing of arguments. Data can be gathered from the user either with the <isindex> tag in the header, which causes a single-

Structural		Descriptive		Visual	
html	document type	a	hypertext link anchor-point	b	bold type
head	document header	code	computer code	br	forced line-break
title	document title	code	computer code	hr	horizontal rule
base	root address for incomplete hypertext references	em	emphasis	i	italics
meta	specification of mapped headers	kbd	keyboard input	tt	typewriter type
link	relationship of document to outside world	samp	sample of input	img	illustrations
isindex	specifies a processable document which can take an argument	strong	strong emphasis		
body	contains all the text	var	program variable		
h1...h6	six levels of section heading				
p	paragraph				
pre	preformatted text				
blockquote	block quotations				
address	addresses				
ol	ordered lists				
ul	unordered lists				
menu	menu lists				
dir	directory lists				
li	list item				
dl	definition lists				
dt	definition list term				
dd	definition list description				
		<b>Form-fill</b>		<b>Obsolete:</b>	
		form	contains a form	listing	use pre
		textarea	free-text entry	xmp	use pre
		input	input field (text, checkbox, radio button, etc)	plaintext	use pre
		select	drop-down menu	nextid	editing control
		option	menu item	dfn	definition of term

**Figure 3.** Markup available in HTML 2.0 (indentation implies the item must occur within the domain of its [non-indented] parent)

line data-entry field to appear, or with the more complex `<form>` element with scrollable text boxes, checkboxes, radio buttons and menus. In this manner, complete front-ends can be manufactured to drive data-retrieval engines of any kind, provided that they operate from the command line, and that the script returns their output in HTML. The user (and the browser) remain unaware that the result has been generated dynamically.

### 2.3 Presentation

HTML is criticised for being ‘presentation-oriented’, but as can be seen from Figure 3, the overwhelming majority of the markup is structural or content-descriptive. However, this does not prevent the naïve or sophisticated author from using or abusing the markup in attempts to coerce browsers into displaying a specific visual instantiation, primarily because none of the browsers (with the partial exceptions of Arena and `w3-mode` for GNU Emacs) performs any form of validation parsing, and will thus display any random assemblage of tags masquerading as HTML. This behaviour has misled even some eminent authorities to dismiss HTML as ‘not being SGML’.

There is thus a conflict between the SGML purist on the one side, who decries any attempt at encoding visual appearance; and the uninformed author on the other, who has been unintentionally misled into thinking that HTML and the Web constitute some kind of glorified networked DTP system.

The purists are few in number but eloquently vocal: however, in general, they acknowledge that visual keys can be included if they are carefully coded. A perceived requirement to allow an author to recommend the centering of an element is thus achieved in HTML3 by the `align="center"` attribute, rather than the unnecessary `<center>` element proposed by the authors of Netscape.

The demands of the author are at their most marked in the approach of publishers and marketing users, who have been accustomed for the last 550 years to exert absolute control over the final appearance of their text. But the Web is not paper, and the freedoms and constraints of the Press do not apply: it is as much a new medium as radio or television. For such an author to insist that she must be able to control the final display to the same extent as on paper is as pointless as insisting that a viewer with a black-and-white television must be able to see the colours in a commercial.

The paradigm has been established that the browser controls the appearance, using the markup as guidelines. There is indeed no reason at all why attributes could not be added so that an author could write

```
<h1 color=green font=LucidaBrightBoldItalic size=24 shading=50>
```

but the user of Lynx or WWW (two popular text-only browsers for terminal screens) would still only see the heading in fixed-width typewriter characters. The habit of insisting that everyone ‘must’ see a particular typographic instantiation is an unfortunate result of a misinterpretation of the objective of the Web: to deliver information in a compact, portable and arbitrarily reprocessible form.

But publishers accustomed to paper, insistent on ‘keeping control’, have of course an entirely valid point, one with which the present author has great sympathy. Why should a carefully-prepared document be made a hames of by a typographically illiterate user who has set `<h2>` to display as 44pt Punk Bold in diagonal purple and green stripes?

The solution probably lies in the implementation of style sheets, perhaps along the lines of those discussed by the authors of Arena[5]. They would in any case only be recommendations: not every user has a CD-ROM of Adobe or Monotype fonts. In any event, if 100% control is essential, as in the display of typographic examples, all graphical browsers can be configured to spawn a window to display PostScript file, although the download time may be a strong disincentive.

It is entirely possible that the control of content will ultimately prove a more attractive option than the control of appearance.

## 3 Publishing with HTML

Setting aside the unresolved questions of display, there are more pressing business problems about publishing on the Web.

The authentication of users is being addressed at several levels, from simple, non-authoritative checks using `identd` to the more complex username-and-password systems employed on some Web pages. From the user’s end, the authentication of the data being accessed is equally important. The openness of the Internet in its raw form allows ‘spoofing’ in both directions, so the emergence of protocols to provide checks is to be welcomed.

The security of network-accessible texts from break-ins remains a concern to anyone providing high-value merchandise, and Web text is in this sense no different from any other computer data. Normal precautions must therefore be taken to prevent theft through other channels (such as remote login), as distinct from theft perpetrated by falsification of Web access.

There is a need for robust solutions to charging and billing for usage, and the secure transmission of financial data, including credit card numbers, digital signatures, and perhaps even EFT transactions. The Secure HTTP (SHTTP) mechanism being marketed by MCom and others is becoming popular as a way of achieving some of this, but the Internet must shed some of its image of lax controls and sloppy housekeeping if it is to achieve sufficient 'respectability' to attract the business of those who are not networking specialists.

The handling of copyright and the intellectual property of electronic texts remains, as ever, an unsolved problem. While copyright law can be used to provide a remedy for breach, the difficulty lies in preventing the breach occurring in the first place. The reason is that (as with other electronic material), copying and reproduction is fast, cheap and easy, once the material is in the hands of the customer. While a supplier may use SHTTP to protect the details of the transaction, once a print file has been sent to someone, the supplier retains no control whatsoever over its use, reuse and abuse. Copies could be sent to dozens others, or printed many times, in the space of minutes.

### 3.1 *Printing from HTML*

The demand for printed copies of Web material is surprisingly high. Although in some cases it is reminiscent of those people who insist on printing their email, it is undeniable that there is a serious requirement for good quality print from Web documents.

Existing solutions to printing SGML text are usually application-specific, being embedded in SGML editors or DTP systems, but there are also some more generic packages:

- Format by Thomas Gordon ( $\LaTeX$ )
- HTMLtOPS by Jan Kårrman (PostScript)
- SGML2TeX and WebSet by Peter Flynn ( $\TeX/\LaTeX$ )
- SimSim by Jonathan Fine ( $\TeX$ )

The use of  $\TeX$  systems for most of these seems to indicate that the similarity of markup concepts has not gone unnoticed by practitioners. The author's own contributions are experimental, but the second of them is planned as an interactive Web service, to be introduced in the summer of 1995. Emailing a URL to the point of service will cause it to be retrieved, typeset, and the output returned to the user by email in PostScript form. As a form of email browser, the control of appearance may lie in the hands of the user, but suggestions for how implement this are currently being sought[2].

### 3.2 *Problems*

Implementing a professional level of typesetting from HTML raises some interesting questions:

- most HTML files are invalid
- most HTML authors don't understand SGML
- most HTML authors couldn't care less
- most World Wide Web users couldn't care less

The handling of missing, damaged or abused tags in a gracious manner is not a feature of most SGML parsers. At the best, a typesetter-browser can only be expected to report to the user that a file is invalid, and while it may be displayed by browsers which do not make any claim to typographic quality, an attempt to make a respectable print job of an invalid file is unlikely to succeed.

## 4 **Development**

The future of the World Wide Web and HTML is uncertain. While development continues, and while new users are anxious to start surfing the net, the existing designs and implementations will suffice. In the longer term, a coalescing of services is likely to occur, but for this to happen, a number of changes need to take place:

- The Web will start to make use of other DTDs, as outlined above. Any file containing a `<!doctype . . . >` at the beginning could cause a browser to retrieve the DTD specified, along with a style sheet, and work much as any SGML-conformant DTP system would.
- Browsers will become pickier, able to offer better services at the expense of rejecting invalid or badly broken files. Arena already performs a form of consistency check on the HTML code of files, and displays 'Bad HTML' in the top corner when an offender is spotted.
- Users will become pickier, demanding better response from the browser, better response from the server, and better facilities from both. As users become more educated about the use of SGML, developers will no longer be able to hide the deficiencies of products under the cover of technical detail.

- This presupposes more user education, which is inevitable in a developing technology. 100 years ago, motor cars appeared on the roads, but few passengers in them understood the use of the levers and rods which controlled them. With some minor exceptions, it is now expected that a driver knows that turning the wheel clockwise turns the car to the right, and *vice versa*. It will not take us that long to perceive the innards of HTML, but it can only be done by training and education.
- At some stage, investment is always needed. Many companies have put substantial sums into the development of Internet resources, and those that have done so with forethought and planning deserve to reap a rich reward. It is a long-term investment, more akin to a partnership, but support is always needed by those who undertake the developments, especially as much of it is done in personal time and at personal expense.

There is still some way to go before we achieve the ease of use of the telephone or the radio, but the path is becoming easier with each new development.

## References

- [1] Berners-Lee T & Connolly D, *HyperText Markup Language Specification — 2.0*, Internet Draft, IETF Working Group on HTML, December 1994.
- [2] Flynn P, *Typographers' Inn*, T<sub>E</sub>X and TUG NEWS, **4**, 1, March 1995.
- [3] Goldfarb C, *The SGML Handbook*, OUP, 1990, ISBN 0-19-853737-9.
- [4] Kernighan BW & Ritchie DM, *The C Programming Language*, Prentice-Hall, 1978.
- [5] Lie H *et al*, *HTML Style sheets*, <http://www.w3.org/hypertext/WWW/Style/>
- [6] Pepper S, *The Whirlwind Guide: SGML tools and vendors*, <ftp://ftp.ifi.uio.no/pub/SGML/SGML-Tools/SGML-Tools.txt>

---

## VIII The Inside Story of Life at Wiley with SGML, L<sup>A</sup>T<sub>E</sub>X and Acrobat

Geeti Granger  
John Wiley & Sons Ltd,  
Baffins Lane,  
Chichester, W. Sussex PO19 1UD

---

### 1 Introduction

As a brief introduction I should say that John Wiley & Sons is a scientific, technical and medical publisher. It is an independent, American family-owned company that was established in 1807, with subsidiaries in Europe, Canada, Australia and Singapore. The European subsidiary opened in London in 1960 and moved to Chichester in 1967 (if folklore is to be believed this was so that the then Managing Director could more easily pursue his love of sailing!).

We publish books, including looseleaf and encyclopaedias, and journals, and most recently electronic versions of some of our printed products. In the future the electronic component of our publishing programme is bound to include products that are only available electronically.

### 2 Setting the Scene

Now to the topic in hand—Portable Documents: Acrobat, SGML and T<sub>E</sub>X. Our association with T<sub>E</sub>X dates back to 1984 when we made the significant decision to install an in-house system for text editing and composition. It was the only software available that wasn't proprietary, which stood a chance of coping with the complex mathematical material we had to set.

As a company we have monitored the progress of SGML since 1985, but have only recently used it in earnest. Our first project is a 5000 page encyclopaedia about Inorganic Chemistry. We rarely get the opportunity to dip our toes in the water—it's straight in at the deep-end! Having said this, we do have a set of generic codes that has been used for a number of years, and everyone is well aware of the principles involved and the value of this approach to coding data.

Adobe Acrobat was launched in June 1993. Our experience of this software dates back a little further than this, because of our links with Professor David Brailsford and the Electronic Publishing Research Group at the University of Nottingham, and their work on the CAJUN (CD-ROM Acrobat Journals Using Networks) project, which we jointly sponsored with Chapman & Hall.

### 3 Complementary not Competitive

The first thing to make clear is that SGML, T<sub>E</sub>X and Acrobat do not compete with each other in any way. SGML is a method of tagging data in a system-independent way. T<sub>E</sub>X is one possible way of preparing this data for presentation on paper, while Acrobat is software capable of delivering data electronically for viewing on screen, or for committing to paper.

From our point of view the fundamental requirement for:

- capturing data
- processing data (text and graphics)
- delivering data (paper/disk/CD/Internet)

is to remain system independent for as long as possible.

SGML, T<sub>E</sub>X and Acrobat achieve this in their part of the whole process. PostScript provides the link that completes the chain.

### 4 SGML in Practice

To describe our experience with SGML I will use the *Encyclopedia of Inorganic Chemistry* as a case study. This encyclopaedia is an 8 volume set made up of 5000 large-format, double-column pages (more than 3 million words). The data consists of approximately 250 articles interspersed with 750 definitions and 750 cross-reference entries. The  
*reprinted from Baskerville*

*Volume 5, Number 2*

text was marked-up and captured using SGML, validated and preprocessed for typesetting. The floating elements (all 2300 figures, 8000 equations, 2000 structures, 1100 schemes and 900 tables) were prepared electronically and delivered as encapsulated PostScript files. Some 150 halftones, about a third of which are colour, complete the data set!

Despite the complex nature of this project, or maybe because of it, we were convinced that using SGML was the right approach. We had to be very sure because this decision presented us with many additional difficulties. Different considerations had to be made at all stages of the production process. (Manufacturing remained untouched.)

Initially, having established the probable requirement for an electronic version, there was the need to justify the use of SGML because of:

- the extra cost involved in data capture
- the different working practices that had to be established
- the project management overhead
- the need to find new suppliers, and the risks that this involved for such a large, high profile project.

#### *4.1 Production Considerations*

This project had an external Managing Editor to commission and receive contributions before it became a live project for us. Once contributions started to arrive it very quickly became apparent that a project management team was needed if this project was to succeed. The initial steps had to be ones of project analysis, determining data flow, deciding who was responsible for what, and ensuring that a progress reporting system was established. It certainly seemed like a military operation at times.

Having made the decision to go with SGML and to ensure that all components were captured electronically we had to find a set of new suppliers. None of our regular suppliers could meet our specifications. Locating potential suppliers was the first hurdle, and then assessing their suitability was the next. Having done this we then had to draw them all together to establish who did what, and who was responsible for what. It had to be a team effort from start to finish and regular progress meetings involving representatives of all parties was the key to an ultimately successful project.

#### *4.2 Problems Encountered*

One of the first considerations was how on earth do we name the files? To ensure portability we set ourselves the restriction of the eight plus three DOS convention. It took some time but we achieved it in the end so you can now identify from the file name the type of text entry, the type of graphics and whether it is single or double column or landscape, and its sequential placement within its type. When you consider the number of files involved, this was no mean feat.

Designing the DTD without all the material available is not the best way to start, but needs must. It meant that some amendments had to be made as the project progressed but none of them proved to be too significant.

Choosing Adobe typefaces, to avoid problems later on, meant that some compromises had to be made. Many people feel that the Adobe version of Times is not as elegant as some.

Also the quality of the typesetting, hyphenation and justification, interword spacing and overall page make-up is not as high as that normally achieved by a dedicated chemistry typesetter.

In addition to the above, we found a bug in Adobe Illustrator! Because the EPS files were being incorporated electronically the accuracy of the bounding-box coordinates was crucial. To cut a long story short they weren't accurate. We spent quite some time establishing the cause of the problem and then had to have a program written to resolve it.

This is not an exhaustive list but I think it will give you a feel for the practical issues involved. Having shared all this with you I should add that all of us involved in the original recommendations remain convinced that it was the right approach. In fact we are now processing two more projects in the same way!

## **5 L<sup>A</sup>T<sub>E</sub>X in Practice**

We've done far too many projects in T<sub>E</sub>X (many in Plain, but a growing number in L<sup>A</sup>T<sub>E</sub>X) to select one as a case study. What I can do is very readily identify the production issues involved in using this software in a commercial environment.

### *5.1 Steps in the Process*

Establishing ourselves as a forward-thinking, progressive company by developing in-house expertise has brought with it certain pressures. In the early days, not only did we have to learn how to use T<sub>E</sub>X, we also had to make it achieve typesetting standards expected of more sophisticated systems. Our colleagues could not see why they should accept

lower standards from us—after all they were paying us (we operate a recharge system so that it doesn't distort the project costing when compared with externally processed projects).

Next came the requests for us to supply style files. Authors knew we used the same software as they did, and wanted to prepare their submission so it looked like the finished product. Some wanted to produce camera-ready copy. In principle this would seem a sensible idea; in fact our commissioning editors, especially those who handle a number of CRC projects, thought it was a brilliant idea. It would save them an immense amount of time and hassle.

Now, preparing style files for in-house use is one thing; preparing them for use by others is something else again. We have to work within strict time and cost constraints, and there are many occasions (dare I admit it?) when we have to resort to, shall we say, less than the most sophisticated way of achieving the required visual result!

When I have attended courses on  $\text{T}_{\text{E}}\text{X}$  and have asked about writing style files the answer has often been along the lines of 'leave it to the professionals'. (I should say it's usually people who make their living in this way who give this response.) This may be fine if a) you can find and afford the professional; b) you don't need to support the file when it is in general use. In our experience the first is difficult to do and the second is an impossibility. The need to support style files cannot be ignored; once they have been provided, no matter on what pre-agreed conditions, queries will arise. It can be very time-consuming, as often queries are not restricted to the style file, but relate to the system being used. It can also take a while to establish the context of the query, resolve it and respond. To meet the expectation that we will support, customise at short notice, resolve technical issues, and communicate via e-mail (preferably responding within the hour) can be difficult, given the level of human resource available.

Once you've got over this initial stage, the practical issues involved in accepting  $\text{L}_{\text{A}}\text{T}_{\text{E}}\text{X}$  submissions can be many. Delivery is the first. Now that we have the ability to receive data electronically our authors cannot understand why we hesitate, and why we still insist on hard copy. Experience tells us that, without hard copy, it is difficult to be sure we have received the final version, and discovering this after a project has been processed is very costly, both in time and money. Any submission that circumvents a stage in the current administration process may drop through a hole and end up taking more time, rather than less, to reach publication. Consideration is being given to this issue, and there is no doubt that in the future electronic delivery will be an acceptable method of submission, but in the meantime everyone has to be patient.

Copy-editing remains a conventional process in the main, although experiments are taking place with copy-editing on disk. This issue is not restricted to  $\text{L}_{\text{A}}\text{T}_{\text{E}}\text{X}$  projects, but the rate of progress is dictated by the ability of our freelance copy-editors to provide this service.

Once you move on to the processing stage the first thing you have to do is find a supplier who is capable of actually processing in this software. This is easier said than done, because it is not considered to be cost-effective by most of our regular suppliers. However, as a result of our persistent requests, some can now provide this service, so we don't have to process all such submissions in-house.

From our own experience we know that producing page proofs is not always straightforward. Over the years we have struggled with amending style files to achieve the correct layout and controlling page make-up. Now that authors are submitting graphics on disk, as well as the text, we are faced with another set of problems. Portability of graphic formats is even more difficult to achieve. I think the number of answers to the question 'When is a PostScript file (or EPS file) not a portable PostScript file?' must be infinite. Even when the content of the file itself is OK, you can still be faced with problems in achieving the required size and position on the page.

Despite all these disadvantages our lives would not be the same without  $\text{L}_{\text{A}}\text{T}_{\text{E}}\text{X}$ , and when compared with processing in other software it can be a real joy! Our archive of projects coded in a form of  $\text{T}_{\text{E}}\text{X}$  will be far easier to reuse than those processed in other software.

## 6 Acrobat at Arm's Length

Although we haven't used Acrobat on a live project in-house yet, we have been closely involved with the development of the EPODD CD. The CAJUN project has been running for well over a year and during this time the complete archive of volumes 1–6 has been converted to PDF, annotated to add PDFmarks and generally massaged into a suitable format for delivery on CD.

As always, the work involved in such a project is more than anticipated at the outset, but it has been an invaluable learning exercise. Being involved in the beta-testing of the software helps you appreciate just how much development work is required for a new piece of software, and although it currently has its limitations the future looks good. Version 2, which is due for release any day now, is much improved, and it is rewarding to see that many of the comments put forward by members of the team have been incorporated.

We are experimenting with small projects in-house to give us a deeper understanding of the practical advantages and limitations of Acrobat. It is easy to get caught up in the euphoria and hype that accompanies the release of a new product, and to overlook the day-to-day difficulties its rapid adoption might bring. Having said this, there is no doubt that it will have a place in our publishing procedures, and may be used in the production cycle for journal articles. Provided that the general administration can cope with the deviation from the norm, supplying author proofs in this way has its attractions. The fact that readers are now freely available and the PDF file can be read on any of the three main platforms is a real boon.

The use of Acrobat for delivering existing print products in an electronic form is one worth considering, especially now that it is possible to integrate it with project-specific software and the security issue has been addressed.

From an inter-company point of view the perceived use of Acrobat for distributing internal documents could again have its attractions. For this to be a real possibility it must be recognised that the use of such procedures is not an innate skill, and so the appropriate level of training and support must be available if it is to be successful.

## **7 Conclusion**

The comments I have made and the case study I have described may leave you with a somewhat negative feeling. I wonder if I have emphasised the problems and not balanced these by identifying the plus points. To put this into context I should say that details of the advantages of any particular approach are usually more readily available, so I have tried to capture a more down-to-earth view.

In reality I am very enthusiastic about the use of SGML, L<sup>A</sup>T<sub>E</sub>X and Acrobat, but am also well aware of what their use in a productive environment can mean. I believe, as do several of my colleagues, that portability of documents is crucial to our ability to deliver data efficiently in a variety of forms, whether this be page-based, highly structured databases or tagged ASCII files. To this end we must be flexible in our approach, and must not be afraid of making investments now that may not bear fruit until some time in the future. This can be a very unnerving decision to make, and for one I am glad it isn't ultimately mine. While I can extol the virtues of a purist's technical approach, obtain the relevant costs and assess the schedule implications, I do not have the entrepreneurial skills required to know when a project is commercially viable (or worth taking a risk on). It is at this point I take my hat off to our commissioning editors, who have the responsibility for turning these experiments into profit for us to reinvest in the next Big Thing!

---

## IX Theory into Practice: working with SGML, PDF and L<sup>A</sup>T<sub>E</sub>X at Elsevier Science

Martin Key  
Elsevier Science Ltd  
m.key@elsevier.co.uk

---

### 1 The Company

While I do not want to make this article a plug for Elsevier, it is first necessary to put our activities into context. Therefore, for those who do not know us, Elsevier Science is part of the Reed Elsevier Group and, in terms of number of journals, is by far the largest publisher of scientific journals in the world. The original Elsevier Company was Dutch based, but now, through acquisition and merger, is an international company with offices in the Netherlands, UK, USA, Switzerland, Eire and the Far East. We publish well over 1,000 scientific, technical and medical journals covering all sections of academe and business.

### 2 The move into electronic publishing

Elsevier's major customers are academic and research institutes throughout the world. Traditionally, academic publishing has relied on authors submitting papers via external academic editors who arrange for the necessary peer reviews. Once accepted, papers are sent to Elsevier for copy-editing, typesetting and compilation into issues. As a result we have in the past received paper manuscripts of varying levels of presentation from around the world. Over the last 10 years it has become apparent that most authors use some form of word processing or computer generated text to prepare their papers. To have these papers typeset means rekeying the manuscript and, what is worse, ending up with electronic files produced by many types of typesetting equipment and software with minimal chance to reuse this material at a later date. For some years the Elsevier Group have been looking at ways to avoid rekeying manuscripts whilst at the same time automating the production process, produce proofs more quickly and create electronic files for multiple use in the foreseeable future.

After many surveys, experiments and discussion groups it was clear that Elsevier should work to accepted international generic standards in order to achieve these goals. The major standards agreed on were Standard Generalised Mark-up Language (SGML) for text, Tagged Image File Format (TIFF), Joint Photographic Experts Group (JPEG) and Encapsulated PostScript (EPS) for graphics and PostScript, and the Portable Document Format, (PDF), also known as Acrobat, for pages. Unlike typesetting codes, SGML does not drive any particular application but can be readily converted to numerous formats for typesetting on paper, database applications, CD Rom and so on. It is therefore an ideal archive medium. TIFF, JPEG and EPS are well documented graphic file formats and are widely supported in terms of external applications. PDF is, perhaps, a risk in that it is the property of a commercial developer (Adobe) but its great flexibility and rapid acceptance by professionals and the academic community, together with the track record of PostScript itself — now a de facto standard — makes its long-term future seem relatively safe. The decision by Adobe to make the Acrobat reader available free-of-charge is another positive sign.

### 3 The concept of Computer Aided Publishing (CAP)

Once the standards were agreed the process known internally as CAP (Computer Aided Publishing) took clearer shape. There are a number of activities which form part of CAP. These include the following: the converting of manuscripts and artwork into electronic files; structuring of text with SGML; editing on screen; automatic proofing; moving and maintaining files on a network; creating SGML (text) and graphic files; receiving PDF files from our typesetters. In addition, a number of journals receive, and use, papers in L<sup>A</sup>T<sub>E</sub>X format which will be discussed later.

### 4 Practicalities: How we do it

CAP started in Elsevier in January 1994, in both Amsterdam and Oxford, with a limited set of journals. The number of journals has been increasing rapidly and in 1995, as software and hardware stabilises, the number of journals is being increased dramatically.

*reprinted from Baskerville*

*Volume 5, Number 2*

The first action when receiving a paper, either on paper or disk, is to log the information on to our production tracking system. All the important details are recorded — title, authors, number and type of graphics, whether it is available on disk etc. This record follows the manuscript throughout its production process and is updated at each stage of its progress through the system. Elsevier encourage authors to submit on disk, and the numbers are rising. If it is on disk it is initially converted to our standard CAP format which allows it to be used by our SGML tagging and editing tool — Pandora — which was developed by staff working in Amsterdam. If it is only available on paper it is either OCR (Optical Character Recognition) scanned and then converted into the CAP format or, if the paper is too complex for scanning, it is keyed by off-shore keying agencies. Whatever the route, it arrives at our Pre-Edit Department in the generic CAP format. Simultaneously graphics are scanned — TIFF for line art and JPEG for half-tones — or redrawn and saved as EPS in some instances.

The text is then tagged using Pandora. The Document Type Definition (DTD) used is the Elsevier DTD (which Elsevier has made publicly available subject to certain conditions) which is fairly complex covering not only text but also tables and mathematics.

After coding and parsing, the text is loaded onto the network server, together with the graphics, using an in-house developed Document Management System which monitors, names and controls the files. As one article can produce more than 20 files, with an average issue of a journal containing 10 articles, the number of files can quickly mount making such management essential. Once the files are on the server, they can be retrieved by the Production Editor who will then edit the article for style, spelling, grammar, etc. and add any additional tags necessary. Graphic files are also checked at this stage to ensure that the correct graphics are linked to the relevant caption. The file is then parsed again to check its validity. Author proofs can then be produced and, once they are received back from the authors and corrections made, the final SGML and graphic files are exported to the typesetter for making up the final pages.

We expect typesetters to retain the validity of the SGML files when producing the pages, and this is strictly monitored. Due to the complexity of the DTD and the relevant inexperience of most typesetters in using precoded SGML files, we have to work with our typesetters quite closely, answering specific queries and offering advice where necessary. However, we do not expect to develop the systems for the typesetters — that is their responsibility. The final, additional requirement we demand from our typesetters is that they supply each individual article, and other elements of the issue, in PDF format. This means that they must have a PostScript setter in order to create these files.

## 5 T<sub>E</sub>X and L<sup>A</sup>T<sub>E</sub>X

In some disciplines T<sub>E</sub>X and L<sup>A</sup>T<sub>E</sub>X are used extensively by authors and, not unnaturally, they would like to submit their articles in this format. Experience has shown that this can be hard work for the Publisher. In some cases, hacking in to such a file to find out how the author's carefully developed macros have been used can be very time-consuming and, in some cases, can take considerably longer than having the paper professionally typeset. However, whenever possible, we will try and use submitted L<sup>A</sup>T<sub>E</sub>X files and, to a lesser extent, plain T<sub>E</sub>X files. However, Elsevier encourage authors to use the Elsevier style file which produce a pre-print type output. This style is then replaced with the journal-specific style file which makes the Publisher's task considerably easier. The Elsevier style files, together with the instruction manual, are available from the three CTAN sites or direct from Elsevier.

L<sup>A</sup>T<sub>E</sub>X has a number of advantages. Pages in camera ready format can be produced readily in-house without recourse to a typesetter, and PDF files can also be generated from the dvi files. Recently, the Production Methods Group at Elsevier Science Ltd has further developed the 'dvi<sub>hps</sub>' converter and L<sup>A</sup>T<sub>E</sub>X macros from the HyperT<sub>E</sub>X project, to fully retain the hypertext links available in the L<sup>A</sup>T<sub>E</sub>X file, as well as generating automatic 'bookmarks' or contents list, directly into the PDF file. In order to meet the full CAP requirements previously mentioned, there is one final part of the equation to be completed — a L<sup>A</sup>T<sub>E</sub>X to SGML conversion. Due to the complexity of the Elsevier DTD this is not a simple task but work is currently taking place to see how far down this road it is possible to go.

## 6 Practical Problems

As with most technical developments there are always problems to be addressed. In the case of CAP they have been surprisingly few. The major problem experienced at an early stage was the lack of SGML editors which could cope with the Elsevier DTD, particularly in the area of tables and mathematics. This problem has been largely resolved by the development of Pandora, a tool which has far exceeded its initial specification as a package which would enable compuscripts to be handled by typesetters. The second problem was one of logistics — how do you train Production Editors to work with SGML on screen editing whilst simultaneously producing journal issues? As previously mentioned, there is also the increased demand we place on typesetters, many of whom have had limited experience

of handling complete journals in SGML. Finally, as Production Editors began to use the DTD in earnest, additional requirements are discovered which means that the DTD must be further developed. As a result, the DTD has become a moving target with more complex requirements being asked for almost daily.

## **7 The Future**

Some people may ask why we are putting ourselves through so much pain. Is it worth it? The market is demanding electronic products in addition to, and sometimes instead of, the traditional paper ones. For those publishers who have tried to use typesetters' tapes for such products, the answer is clear. The availability of generic coded data which can be manipulated in multifarious ways is clearly the route to take. In addition to meeting the demands of our market, we are also satisfying the demands of our producers — the authors — who create 'electronic' versions of their articles and who naturally expect that we, the Publishers, should be able to use them. Finally, the Production process itself is being streamlined allowing for more efficient and faster production times.

---

## X SGML and L<sup>A</sup>T<sub>E</sub>X

Horst Szillat

szillat@berlin.snafu.de

---

SGML — *Standard Generalized Markup Language* — is a formal language to describe structured text documents. It should be introduced here by comparison to T<sub>E</sub>X and L<sup>A</sup>T<sub>E</sub>X.

It is interesting to have a look at how Donald E. Knuth introduces T<sub>E</sub>X in the T<sub>E</sub>Xbook himself. The beginning is to simply type in the text and T<sub>E</sub>X mainly does what one expects it to do. Quite a lot of more or less complex rules have been implemented to provide these results. An example of this behaviour is the *space factor code* (`\sfcode`). Using this code T<sub>E</sub>X is able to identify most of the ends of sentences. Moreover T<sub>E</sub>X is realized in a way that one can program almost all kinds of printing layouts. In this way one can program a macro which influences the layout in any place. So T<sub>E</sub>X is a layout oriented system which is able to format texts for printing and to do a bit more.

Although L<sup>A</sup>T<sub>E</sub>X simply is T<sub>E</sub>X, too, and has all these characteristics, too, it introduces a new idea of representing the text input. The basic idea is that a text is given in the form of *embedded environments*. The layout of a text portion depends on the environment it is embedded in. Moreover, the layout of whole environments may depend on which other environment they are embedded in. The user can define new environments (`\newenvironment`) which realize a user defined layout. But the main point is that the author inputs his text on a less technical but a more abstract level. This way L<sup>A</sup>T<sub>E</sub>X enforces the idea of separating the text structure from the printing layout. Changing the layout in L<sup>A</sup>T<sub>E</sub>X means to replace the existing style files, only. One could do the same in plainT<sub>E</sub>X directly, of course. One can do structured programming in assembler, too, but assembler does not enforce it.

Now one can simply say SGML is L<sup>A</sup>T<sub>E</sub>X without T<sub>E</sub>X to be written in a slightly different manner. This means SGML is a representation of the text in its hierarchical structure without any idea of a layout. If one has lost the layout there has to be an advantage on the side of the text structuring. And so it is, indeed. L<sup>A</sup>T<sub>E</sub>X's environments are called *elements* in SGML. Within a certain model one can now define which way the elements are embedded in each other and where text is to be allowed. Within that model the amount and the order of embedded elements and text is defined.

Such a definition of a text structure is called *document type definition (DTD)*. The “best-known” example of a SGML document type definition is HTML (*Hypertext Markup Language*) used for the World Wide Web. While processing the document an SGML-parser is able to validate the structure of the document by the given document type definition. A simple example should illustrate this:

```
<!ELEMENT section - - (paragraph?,subsection+)>
<!ELEMENT subsection - - (paragraph,paragraph+)>
<!ELEMENT paragraph - - (#PCDATA)>
```

These lines are to be read as follows: An environment/element called `section` consists of maximum one `paragraph` and at least one `subsection` in this order. A `subsection` consists of exactly one `paragraph` plus at least one `paragraph`, e.g. at least two `paragraph`s. And at last, a `paragraph` consists of letters. Here it is not possible anymore — unlike in L<sup>A</sup>T<sub>E</sub>X — to put the first `subsection` before the first `section`. One could define the L<sup>A</sup>T<sub>E</sub>X environments with such control structures, too. But again, L<sup>A</sup>T<sub>E</sub>X is not designed for this goal and does not enforce it, while such validating is the nature of SGML.

Another structural advantage over L<sup>A</sup>T<sub>E</sub>X is the consequent distinction between *parameter* and *data*. The lines

```
\label{Hallo!}
\section{Errors}
\unknown{whatever}
```

show that in L<sup>A</sup>T<sub>E</sub>X one can never be sure what is human readable text (*data*) and what is internal technical information (*parameter*). On the other hand SGML has a strict idea of this distinction. As long as the SGML structures are not mis-used malevolently it is possible to make this distinction without even understanding the content. This is an important condition for any computer based data processing. An example will be given later.

But even in the days of total computerizing the final goal of text representing is to print the text onto paper. There are two projects/tools specially designed for the printing of SGML documents. FOSI (Formatted Output Specification Instance) and DSSSL (Document Style Semantics and Specification Language). But why not use L<sup>A</sup>T<sub>E</sub>X? L<sup>A</sup>T<sub>E</sub>X has some characteristics which make it the first choice.

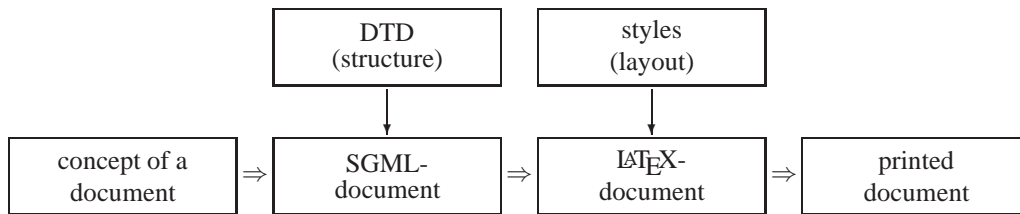


Figure 1. Processing of a SGML document

- The structure of SGML and L<sup>A</sup>T<sub>E</sub>X are very close, so that the documents are easily to convert.
- L<sup>A</sup>T<sub>E</sub>X is a programming language and therefore can realize a wide range of unforeseen layouts.
- L<sup>A</sup>T<sub>E</sub>X has been used for many years by a large number of people. So there exists a widespread experience.

A principal scheme of the processing might look as shown Figure 1.

Unfortunately it is not sufficient to convert the elements into environments and to write the needed style files. As already mentioned SGML and L<sup>A</sup>T<sub>E</sub>X have different ideas of what is data and parameters. So it is especially necessary to transform SGML-data to L<sup>A</sup>T<sub>E</sub>X-parameter so that L<sup>A</sup>T<sub>E</sub>X can handle it more flexibly. A typical example is the following:

```

<section label="main-section">
<title>section title</title>
section content
</section>

```

What one would like to get is something like this:

```

\section{section title}\label{main-section}
section content

```

One should note that `main-section` is a parameter before as well as after conversion while `section title` moves from being data to being a parameter. The easiest way to solve this problem is to introduce additional braces within the L<sup>A</sup>T<sub>E</sub>X environment. Depending on the number of parameters defined in the definition of the environment the data is treated as a parameter or the last parameter is treated as data:

```

<name parameter="value">data</name>

```

converts to

```

\Bsgml{name}{value}{%
data%
}\Esgml{name}

```

With some (yet still to be defined) command

```

\NewSgmlEnv{name}[n]{...}

```

one gets:

- both `value` and `data` being a parameter for  $n = 2$ .
- `value` being a parameter and `data` being data within the environment for  $n = 1$ .
- both `value` and `data` being data within the environment for  $n = 0$ .

Note that this conversion can be done without any conversion parameters. All programming, e.g. replacements are done in L<sup>A</sup>T<sub>E</sub>X. This is a major difference to the widely used SGML-to-whatever converter `format` which works with replacement tables.

But the real reason for why I started to develop my own SGML to L<sup>A</sup>T<sub>E</sub>X converter is that I felt the necessity to manipulate the data within the conversion process.

The main questions are what information about the used words are needed for typesetting and where this information comes from. Again this seems to be a typical non-English problem. In German there are two similar problems: hyphenation and (wrong) ligatures.

Basically German hyphenation rules are easily to be adapted for pattern matching and ligatures can be applied. (Hyphenation is allowed before the last consonant out of a group of consonants. There is no hyphenation within a group of consonants at the very beginning or end of a word. Certain combinations of consonants count as one single consonant. Easy, isn't it?) At the present there is a problem with the umlauts. But this problem should disappear with the `dc`-fonts. The real problem raises with complex words, e.g. words which are composed of several words but

look like one. These words have to be hyphenated between the elements of the compound. This fools every pattern matching. Moreover, there should not be any ligature in these places. The reason is that one does not want to have less space “between words”.

An example of a rather unsuspecting word is `aufflammen`. One would guess the hyphenation `auff\lam\men`, which is wrong, of course. The English translation gives a hint: *flame up*. Within terms of `german.sty` one should write `auf" |flam\men`, where `" |` means: hyphenation is allowed but no ligature is allowed. The printing result is “aufflammen” instead of “aufflammen”.

Unfortunately  $\TeX$  is unable to store this information neither in the hyphenation table nor in the document preamble by `\hyphenation`. Maybe a successor of  $\TeX$  will be able to do so. So far an author writing in  $\LaTeX$  has to input this information directly into the document, well — if he cares. . .

Using a conversion from SGML to  $\LaTeX$  the converter would be the right place to insert the additional hyphenation and ligature information. The converter has to use two dictionaries — a standard dictionary and a special dictionary. It is not unusual that special matters need special terms and consequently special dictionaries. But in German the problem is that one can create new complex words ad hoc. These new compounds may be specific to a particular document. So it would be a nice idea to ship this special dictionary as a structural part of the document!

In this way the author does not have to care about every single hyphenation and ligature exception, but additionally has a spell checker.

But unfortunately there is even a worse case which needs special treatment. It is the word `Baumast`, which can be `Bau\mast` (*mast used in building*) `Baum\ast` (*bough of a tree*), both made of wood, of course. This is a really rare case that a word must be tagged with an additional information where it occurs within the document. This information should explain which word is to be meant. One could do that in the form of an explicit hyphenation information. In SGML it could look like

```
<word which="Bau\ast">Baumast</word>
```

(This example is simplified. It would be more correct to use a SDATA-Entity so that the  $\LaTeX$ -specifics are hidden.)

Note that the hyphenation information on the words `aufflammen` and `Baumast` are totally different things. The first one is part of the layout information (how to print out?), while the second one is a structural part of the document (which word?).

Summarizing one can state that SGML and  $\LaTeX$  are a good pair. Using the specifics of both systems one can do a lot of things correctly in an easier way.

#### 0.1 Further reading

- H. Szillat: *SGML — Eine praktische Einführung* ISBN 3-929821-75-3, Int. Thomson Publ.
- ftp-server: `ftp.ifi.uio.no`
- news groups: `comp.text.sgml`, `sgml-l`

---

## XI Questions and Answers

compiled by Jonathan Fine

---

The last session of the Bridewell meeting was a panel comprising the speakers David Barron [DB] and Martin Key [MK], joined by Lou Burnard [LB] (Oxford, Text Encoding Initiative) and David Evans [DE] (part of David Brailsford's group in Nottingham). The session was chaired by David Penfold [DP] from the co-sponsors, the BCS Electronic Publishing Special Group.

(I have prepared this report from a not always audible tape recording. Remarks have been edited. I hope I have not introduced any error or misrepresentation. Questions have been set in italics.—Jonathan Fine)

*What solutions are there going from one encoding method to another, say from Microsoft to L<sup>A</sup>T<sub>E</sub>X?*

[LB] I recommend via SGML as internal format. The public domain tool Rainbow Makers has an interesting approach. It takes a document marked up with formatting codes, and turns those codes into SGML tags. So you do get terrible things such as tags for font and type-size, but now they are represented using the SGML format. Translating from that to real SGML is a lot easier.

[DP] *Word-for-Word* has been well used in the publishing industry for years. It converts between stacks of mainly word-processor (but also Frame for example) formats. I expect they will eventually get round to SGML.

[Malcolm Clark] *If all I'm interested in is portable documents, in other words shifting a document from one site to another electronically, why don't I just standardise on Microsoft Word? It's not high enough quality for publishing, but for 90% of what I do, memos and stuff like that, I do it in Word and attach it to the electronic mail message I am sending. The recipient at a Windows or Macintosh unbundles it and they've got it in the same format I created it in. It's solved all the problems.*

[LB] Because then you can only talk to people who've done the same thing.

[MK] Several answers. As a company, Elsevier wishes to retain the material they produce for some considerable time. We still sell material that is 10 or 20 years old. Microsoft Word as a format is fine for sending it off today, but in 10 years time, who knows? It probably won't be compatible with anything. So retaining it in that format is no use. We have to convert it into something we can do things with. Secondly, there are limitations with what you can do with a Word document. Such as how you can search text, specifying where the author's name is, etc. SGML allows us to structure the complete document properly.

[DB] There is a distinction between portability for immediate delivery and portability for archiving (see his article). Also, Warwick University (MC's location) must be different from Southampton, where no such uniformity exists. For example computer scientists use T<sub>E</sub>X. The problem is like herding sheep to get them to move in the same direction.

[Allan Reese] Word is not a standard, it's a mess. At our site we have different versions of Word on different platforms, and they have different and incompatible document formats. The lowest common denominator of portability is to have the text transmitted from one place to another. With the Internet this is generally not possible. As soon as you are using a particular character set lying outside ASCII you are lost. One example is a text file (produced by a software company) which was transferred from Mac Word to Win Word without being checked carefully. The left quote character had come out as a 'O' slash. These funny things happen. Another example is from Spain, sent to a news group. The sender has the character 'ñ' on his keyboard. He presses this key and it comes out as 'n' on the screen. He sends it to me and it comes out as '\$'. He can't even send the word 'Español' to the Spanish language news group! Transmission of text is a big problem which a lot of users haven't yet tackled. In academic publishing one will have to deal with multiple character sets, if only to be to accommodate authors' names.

[Jonathan Fine] *Two related questions. Firstly, where will we be in the year 2000? If we have a meeting here in five years time, what will have happened? Secondly, if SGML succeeds, what will fail?*

[LB] In five years time people will be talking about Microsoft in rather the same way we talk about IBM. Remember IBM, they used to make computers. What will fail will be the forces of the evil empire. Namely the idea that it is perfectly legal and correct for any software company or equipment vendor to take information away from the people who have created it and lock it up in a proprietary format. That is an idea that I really would like to see the end of. (Others expressed doubt at the early demise of Microsoft.)

[MK] As a publisher, still dependent on paper. Five years is not very long. Unless an electronic product appears that is really user friendly to read for any length of time, we will still want paper. There will be more electronic products, particularly on CD-ROM. In our environment a lot of specific document formats will probably fail. Ventura Publisher for example. People will concentrate on just three or four products eventually. L<sup>A</sup>T<sub>E</sub>X and T<sub>E</sub>X will still be around, and a few word processors.

[Sebastian Rahtz] *A slightly heretical question about maths and chemistry. A lot of effort has gone into providing DTDs for these things. Perhaps these will just wither and die. In five years time perhaps we will stop pretending that math is structured and regard it in the same way as we regard graphics? Would any one like to defend the SGML markup of maths?*

[MK] The only reason it is useful, is that it is independent of fonts. With L<sup>A</sup>T<sub>E</sub>X you are still dependent on the font. When we combine different articles into a book, we want a uniform appearance. We don't want a mixture of fonts, otherwise we're back to the horrible camera ready copy. As for the fact that SGML maths is structured, I wouldn't particularly want to defend it.

There was quite some discussion of this from the floor, which the tape recorder did not clearly pick up.

[Dina Desai] *We would like to use SGML markup for our maths. What would you suggest?*

[DB] Do you mean what DTD, or what software, or both? *What DTD?*

(Some information given by Mike Popham about specific math DTDs.)

[Gerard van Nes] The whole problem with maths and SGML is that we simply need an SGML editing program which is able to display as we write such complicated formulae. It is of course very uncomfortable to write the huge amount of mark-up as one needs for SGML maths. But if you have a really good SGML editor it's no problem at all.

[DB] I would have for maths a single tag, which says `<maths>` with an attribute which is notation, whose values will obviously be `tex`, `eqn`, etc. If you know it's `eqn` or whatever, it is searchable. You can put hyperlinks into it.

*About DTDs. About compound documents really. Say someone wants a journal with a video snip of the author explaining the article, a sound bite, or what have you. Where would one get a DTD for this?*

[DB] Much the same as the maths. A tag that says `<sound>` and an attribute which says which encoding scheme you are using.

[LB] There is an application of SGML called HyTime which is (about to become?) an ISO Standard. It defines time-based media of all kinds and also different architectures within which you can associate events happening in time. There is one product that can function against HyTime specifications, it is something to watch out for. There is in the TEI Guidelines a simplified version of some HyTime concepts.

[DB] In the latest issue of EPodd there is a paper with the title *Why Use HyTime?*

[Angus Duggan] *Adobe put a lot of effort into Acrobat as a static encapsulation format, able to reproduce the exact form of documents. Will this in the future be important, or will content be all? Will the first published form of the document be important?*

[DE] For archival purposes things like Acrobat are very useful, because they can encapsulate exactly how a document looked. For other things, such as database access and searching it is the content which is more important. So you might want two different electronic forms of the document. Also, the printed and on-screen versions of the document might be formatted differently.

[Angus Duggan] *If we want to represent content and we do want keep the original form it was published in, then obviously Acrobat solves the one problem and SGML the other. What thoughts about document formats which maintain both equally as well?*

[LB] I'd like to question one of your premises. You talk about the original form of the document. I think we're going to forget what that is very soon. I don't know what the original form of the TEI guidelines as published is. It was produced as an SGML document. At home it is white letters on a black background. In the office black on a white. In yellow on green when I was in Chicago. This was the authoring. Similar remarks apply to the printing, on US and UK sized paper. We had to do some fiddling around with the page numbers, as you can imagine. There is another version of the guidelines which is equally authoritative and has exactly the same content, and that exists on a DynaText screen. It has no page numbers at all. I'm trying to make the point that I don't know which is the original form. They're all equally valid.

[MK] When authors have their references in the article they often refer to an article in a book by its page number. How on earth will you make a reference to a location in an electronic document which does not have page numbers?

[LB] By referring to the logical organisation of the text. Paragraph 38 within division 3 of etc.

[MK] I'm sure that we will eventually have a combined product which will have the format and all the structure in it from the SGML. All in one product. Because there's so much more you can do with that than you can with just PDF. The problem, as David Brailsford mentioned this morning, is the Brand-X between the SGML and the PDF. Until we can resolve that problem we can get to PDF, but not via the generic route, which is what we as publishers want.

[Allan Reese] About chasing up a reference. Page numbers are physical objects, and when the document changes the physical indexing is out of date. With electronic documents you will go just by keyword and content indexing. You won't have to know where it is. You just say I want the paper by Fred Smith or whoever.

[MK] That works when you are searching an electronic product from another electronic product. It won't work from a paper to an electronic product.

[Angus Duggan] Intermediate version of documents. If people are publishing on the Internet, if you put a content link in a document to a document being revised, this may change or break the link. So you need to have links to particular versions of particular documents.

[David Coton] Chapters and verses are a menace. We are looking at how to regard text in an object-oriented way. One big problem with such a scheme is that Bible text has two hierarchical structures. It has chapter - verse structure, and it has section - paragraph - sentence structure. Both are useful, both are valid. Both are in different circumstances necessary. But the two do not coincide. Any object-oriented system has difficulty with that. There are ways round it. We are looking at introducing small enough units of text to give coincident boundaries. SGML also has a problem with this. I'm told that the standard allows for dual hierarchies. However, none of the existing tools implement it.

[DB] There is only one product I've seen that supports this CONCUR feature, which is the Mark-it parser, which is quite old. This problem is discussed in the TEI guidelines. I don't think this is an SGML problem. It is a characteristic of textual materials that they can be organised in many different ways. This is inherent in the nature of text.

[Jonathan Fine] *I'm not sure I should be asking this question. What future for typography?*

[MK] If you believe the formatted original will continue, not replaced by a large amorphous glob of text, there still is a case for typography. It is very important to read something, to understand how it's put together. Reading off a screen is difficult anyway. Typography is purely there to allow you to read something easily and quickly. I can't see any reason why it should disappear. Especially as I still believe paper will be around for a while.

[LB] I think that typography is hard, hard, difficult and very important. There are so many people clamouring in the Web and other electronic marketplaces, that anything that stands out will be enormously important. One of the skills conspicuously missing on the Web is good typographic understanding. The skills needed to present stuff effectively and well in the electronic medium are really an outgrowth from the skills the typographers have developed in the past.

[DP] Typography is one aspect, but information design is perhaps an even more important. We've hardly covered information

design. If more and more people are putting things on the Web and not thinking about how they design the document, then the morass of information overload is going to get worse and worse.

[**DB**] If you go to `www.whitehouse.gov` you'll find that it has been done by a professional graphics designer. That really makes it stand out on the Web.

[**Mary Dyson**] Can I add to that? Typography on the screen. Have we got it well sussed? I don't think we have yet. To go back to the old chestnut, it's not necessarily just transferring the same principles. Italics are not terribly good for emphasis on screen. So you want perceptual equivalents of legibility.

[**DP**] We've come to a good point to stop. One final thing. The Web assumes one model of access to information. There are many others, such as browsing, which are virtually impossible on the Web. Maybe we should be thinking about how other forms of access to information are possible. Can I thank the four people on the panel and everyone else, particularly the speakers.

---

## XII Maths in L<sup>A</sup>T<sub>E</sub>X: Part 4, Numbered and Unnumbered Things

R. A. Bailey  
Queen Mary and Westfield College,  
University of London

---

### Recall

This is the fourth in a sequence of tutorials on typesetting Mathematics in L<sup>A</sup>T<sub>E</sub>X. The first three appeared in issues 4.4, 4.5 and 5.1 of *Baskerville*. The series includes some things which can be found in [4], but I am working in more things which, while straightforward and necessary for Mathematical work, are not in [4] or [5]. In this tutorial I concentrate not on Mathematical formulae but on things like equations and theorems which Mathematicians like to display in special ways and to number (or not).

In case you missed the first three tutorials, I remind you that I expect you, the reader, to do some work. Every so often comes a group of exercises, which you are supposed to do. Use L<sup>A</sup>T<sub>E</sub>X to typeset everything in the exercise except sentences in italics, which are instructions. If you are not satisfied that you can do the exercise, then tell me. Either write to me at

School of Mathematical Sciences  
Queen Mary and Westfield College  
Mile End Road  
London E1 4NS

with hard copy of your input and output, or email me at `r.a.bailey@qmw.ac.uk` with a copy of the smallest possible piece of L<sup>A</sup>T<sub>E</sub>X input file that contains your attempt at the answer. In either case I will include a solution in the following issue of *Baskerville*: you will remain anonymous if you wish.

### Answers

I promised to answer all questions arising from this series of articles (as far as I can).

#### *Size of parentheses*

Charles M. Goldie asks why I put  $(t^{g^{-1}})_v$  in Exercise 26 instead of using `\bigl(` and `\bigr)` to make the parentheses larger than what they enclose. The answer is that you would need `\Bigl(` and `\Bigr)` to make them large enough in this case: I should have used `\left` and `\right` (see Part 2 of this series) but I was lazy.

Both he and Charles Leedham-Green have asked why I permit, or even encourage, deeply nested parentheses, as in

$$u(P((1 - \varepsilon)z)), \tag{1}$$

without using commands like `\bigl(` to make some of the outer parentheses somewhat larger. In Chapter 17 of [3], Knuth advises that authors should use `\bigl` and its relatives to specify the size of parentheses and other expandable fences, to improve the readability of their formulae. However, I have deliberately avoided telling people about these commands.

I have two reasons for ignoring Knuth's advice. The first is that I regard L<sup>A</sup>T<sub>E</sub>X primarily as a system for *authors*, not for typesetters. Authors should not be stopping to worry about the size of parentheses, particularly if the level of nesting may change in a later version of the document. Conventions on size of fence should be a matter for the style designer, not the author. If someone can write a style file that automatically detects the level of nesting and adjusts the size accordingly, well and good. I have no objection to using such a style file; I do object to interrupting my Mathematical thoughts to fret over niceties of sizing.

My second reason is related to the first. Journal editors and executive editors tend to have policies about the size of fences, and they will impose these policies whatever we as authors do. So it is generally a *complete waste of time* for authors to use `\Bigr` and the like, or for referees to insist on them.

This is not to say that I disapprove of `\left` and `\right`. These commands automatically adjust the size of the fence to fit what is inside it. Adding, say, an extra item inside a `\left( ... \right)` does not cause the author

to rethink the size of the parentheses. In fact, in my basic style file I have macros such as `\probab`, `\setof` and `\card` which use `\left` and `\right` precisely so that I can type as I think, *the set of ...* rather than *squiggly brackets, now what size and how much space?* (Oh, all right:

```
\newcommand{\card}[1]{\left|#1\right|}
```

—you can guess the others.) In fact, `\left` and `\right` make no difference to the formula in (1), so neither of my questioners will be satisfied by my answer.

#### *Interchangeability of parentheses, brackets and braces*

Charles M. Goldie also asks if I have an opinion about whether nested parentheses should be routinely replaced so that one uses the sequence `{[(...)]}`, which is demanded by some journals. I do have an opinion, quite a strong one, probably because one of the journals in which I publish most frequently insists on the sequence `{[(...)]}` and shows surprise (or the executive editor does) each time that I explain that I am using `{...}` to denote a *set*. My opinion has been admirably expressed by Ellen Swanson in her bible of Mathematical typesetting [6]:

Often, however, the author of research mathematics attaches a special meaning to different types of enclosures, and this author believes that they *should be left in whatever order and variety the author has indicated in the manuscript.*

(her italics).

## 5 Numbered and Unnumbered Displays

### 5.1 Unnumbered Maths displays

Use `\[` and `\]` for an unnumbered single line of displayed Maths: see Part 1. If you have two or more lines of displayed Maths that must be vertically aligned then you need one of the `array` environments. I shall deal with them in the final tutorial in this series.

### 5.2 Unnumbered word displays

Sometimes what you want to display is not simply a formula but a verbal condition that may or may not involve short pieces of notation. For example:

each basis vector  $f$  in  $V_{T,B}$  is orthogonal to every basis vector in  $V_{B,T}$  except  $f\psi$ .

If this will fit on a single line then you can use `\[\mbox{...}\]`, but this is not very satisfactory because you have to stop and think how long it is and it is subject to changes in the line width. I find that the `quote` environment works well for such displays.

### 5.3 Numbered Maths displays

Use the `equation` environment for a numbered single line of displayed Maths such as (1) in the ‘Answers’ section above. If you have two or more consecutive equations or formulae that do not need to be vertically aligned, simply use one `equation` environment per line. For vertical alignment, wait until the tutorial on arrays.

If you put a `\label` within an `equation` you can painlessly refer back (or forward) to that equation.

```
For contrasts, we put
\begin{equation}
W_T = V_T \cap V_0^\perp
\label{contrasts}
\end{equation}
The space~ $W_T$  was defined
in Equation~(\ref{contrasts}).
```

For contrasts, we put

$$W_T = V_T \cap V_0^\perp \tag{2}$$

The space  $W_T$  was defined in Equation (2).

### 5.4 Numbered word displays

Sometimes word displays also need to be numbered for reference.  $\LaTeX$  does not directly provide an environment for this, but I find that the following works quite well.

```
There is a bijection ...
\begin{equation}
\begin{minipage}[t]{0.8\linewidth}
```

```
each basis vector~$f$ ...
\end{minipage}
\end{equation}
Using ...
```

There is a bijection  $\psi$  between the bases which satisfies:

$$\text{each basis vector } f \text{ in } V_{T,B} \text{ is orthogonal to every basis vector in } V_{B,T} \text{ except } f\psi. \quad (3)$$

Using  $\psi$ , we can show that ...

These displays are numbered in the same sequence as the equations, and can be labelled and referred to in just the same way. Note that I have made no attempt to make the indentation the same as that in quote.

You can suppress the [t] if you want the number to be vertically centred on the display. You can replace 0.8 by any reasonable fraction. There is a catch, however: if you have numbered word displays labelled (9) and (10) you may find that the second one comes out shifted to the left, to allow space for the wider label. Whether or not this happens depends on the settings of other parameters, such as `\linewidth`. With the default width for A4 paper in 10pt in L<sup>A</sup>T<sub>E</sub>X 2<sub>ε</sub>, I found that I had to decrease 0.8 to 0.75 in order to have satisfactory word displays numbered (9) and (10).

Of course, if you have two or more such displays you should make an environment for them. I do it as follows.

```
\newenvironment{condition}%
{\equation%
 \begin{minipage}[t]{0.8\linewidth}}%
{\end{minipage}\endequation}
```

You may wonder why I have used `\equation` and `\endequation` in the definition instead of `\begin{equation}` and `\end{equation}`. This is because of the clever things that L<sup>A</sup>T<sub>E</sub>X does with spaces before and after displayed material. When you type the line

```
\end{equation}
```

L<sup>A</sup>T<sub>E</sub>X ignores the spaces on the rest of the line; if you type the line

```
\end{condition}
```

and the final part of the `condition` environment is `\end{equation}` then this forgetfulness about spaces is not passed through to `\end{condition}`. Use of the more primitive `\equation` and `\endequation` does pass on the forgetfulness.

### 5.5 Numbering equations within sections

By default, equations are numbered 1, 2, ... right through the document in the `article` class. To make them numbered within sections you need

```
\renewcommand{\theequation}%
{\thesection.\arabic{equation}}
```

Then the first equation in Section 1 will be numbered 1.1, the next 1.2, and so on. However, if there are four equations in Section 1, then the first equation in Section 2 will be numbered 2.5 because the `equation` counter has not been reset at the start of the new section. To correct this, you also need

```
\@addtoreset{equation}{section}
```

Because of the @ sign in this command, you must either place it in a style file or make sure that it comes between the commands `\makeatletter` and `\makeatother` in the preamble to the document.

### 5.6 One-off numbering of equations

Occasionally you want to number an equation not in the main sequence but by a particular symbol, such as (\*) or (1.1'). Use the following `oneoff` environment in place of `equation`, putting the desired symbol as the single parameter.

```
\newenvironment{oneoff}[1]{\equation%
 \addtocounter{equation}{-1}%
 \renewcommand{\theequation}{\mbox{#1}}}%
{\endequation}
```

For example,

```
\begin{oneoff}{*$}$
a(p_i,q) - a(p_j,q) = 0 \bmod s
\end{oneoff}
```

$$a(p_i, q) - a(p_j, q) = 0 \pmod s \quad (*)$$

(See [4, page 92] or [5, pages 98–99] for how these counter commands work.) Note that automatic cross-referencing does not work for such equations.

If you want a one-off equation numbered 1.1' related to Equation (1.1) then give the latter a label (say, rowsum) and then do

```
\begin{oneoff}{\ref{rowsum}$' $} ...
```

### 5.7 Subsequences of equations

Suppose that between Equations (5) and (7) you want a sequence of equations numbered (6a), (6b) etc. Put the following in the preamble to the document (or in the style file).

```
\newsavebox{\saveeqn}
\newcounter{subeqnno}
\renewcommand{\thesubeqnno}{\alph{subeqnno}}
\newenvironment{subequations}%
  {\refstepcounter{equation}%
   \savebox{\saveeqn}{\theequation}%
   \setcounter{subeqnno}{0}}%
  {}
\newenvironment{subeqn}%
  {\refstepcounter{subeqnno}%
   \oneoff{\usebox{\saveeqn}\thesubeqnno}}%
  {\endoneoff}
```

(See [4, page 101] or [5, pages 107–108] for details of `\savebox`.) Then use `subeqn` in place of `equation` for each of the equations (6a), (6b) etc., and place the whole subsequence in the `subequations` environment.

Automatic cross-referencing doesn't work for these either. The reason is that in constructing `oneoff` and `subeqn` I have *used* the `equation` environment rather than *mimicking* it, with the result that any `\label` picks up the equation counter. To do this properly you would have to copy out the `equation` part of `latex.tex` (which is well documented) and hack it (for 2.09ers; of course, L<sup>A</sup>T<sub>E</sub>X 2<sub>ε</sub> persons would have to hack part of `lmath.dtx` and `classes.dtx`, which some would argue are even better documented). I have never needed this construct often enough to bother to do it properly, but I am sure that it could be done.

## Exercises

**Exercise 48** Make a numbered displayed equation saying

$$t(vP_g) = (t^{g^{-1}})v$$

and a sentence which refers to it.

**Exercise 49** Make an unnumbered word display saying

There is a natural surjective homomorphism  $\phi: G \rightarrow G/N$  with  $\ker(\phi) \simeq \text{Im}(\phi)$ .

**Exercise 50** Make a displayed numbered verbal condition saying

for all  $A, B, C$  in  $P$ : if  $A \prec B$  and  $B \prec C$  then  $A \prec C$ ; and if  $A \preceq B$  and  $B \preceq A$  then  $A = B$ .

Then add a sentence which refers to it.

## 6 Theorems and their friends

### 6.1 Basics

To make a new environment called `thm` for theorems, do

```
\newtheorem{thm}{Theorem}
```

This sets up the environment, which is then used as follows.

```
\begin{thm}
The kernel of a homomorphism
is a congruence.
```

```
\label{basic}
\end{thm}
In Theorem~\ref{basic} we ...
```

**Theorem 1** *The kernel of a homomorphism is a congruence.*

In Theorem 1 we ...

The theorems are all given the heading ‘Theorem’. They are numbered automatically, and may be cross-referred to in the usual way.

For clarity in the rest of this section, I shall call the item like `thm` the *theorem environment*, the item like `Theorem` the *theorem name*, the text like ‘Theorem 1’ the *theorem head*, and the text like ‘The kernel of ...’ the *theorem body*.

Note that there is nothing to prevent two different theorem environments having the same theorem name. Indeed, the theorem name can be empty.

By default, the theorem head is in bold and the theorem body is in italics. The theorems are numbered in arabic numbers, in a single sequence throughout the document (in the `article` class). All of these defaults can be changed, as I show below.

It may not be obvious to the novice user, but there is more to an environment created with `\newtheorem` than special layout and automatic numbering. The spacing before and after each theorem environment is controlled, and penalties are set so that no page break will come after the first line of a theorem environment unless there is a natural break-point in the text.

Unfortunately, there is a bug (oops, feature) in L<sup>A</sup>T<sub>E</sub>X which means that if you put `\label{...}` immediately after `\begin{thm}` you spoil this page-breaking penalty. But the `\label` should be somewhere easy to find, so I always play safe and put it immediately before `\end{thm}`.

## 6.2 Named theorems

If you have a theorem environment `thm` then you can use an optional argument to `thm` to obtain a named theorem. For example,

```
\begin{thm}[The Central Limit Theorem]
If  $X_1$ , ...
```

**Theorem 2 (The Central Limit Theorem)** *If  $X_1, X_2, \dots, X_n$  are independent ...*

or

```
\begin{thm}[Galois, 1832]
```

**Theorem 3 (Galois, 1832)** *If  $L : K$  is a finite normal ...*

## 6.3 Sequences of numbering

Two optional arguments to `\newtheorem` give you control of which theorem environments are numbered in which sequences. Although it is logical to number theorems, lemmas, corollaries etc. all in their own sequences, it is much easier to find your way around a long document if they are all in a single sequence. To get a theorem environment `lem` numbered in the same sequence as `thm`, do

```
\newtheorem{lem}[thm]{Lemma}
```

After the theorems we have had so far, if we now do

```
\begin{lem}
With the above notation ...
\end{lem}
```

we get

**Lemma 4** *With the above notation ...*

The other optional argument numbers the theorem environment inside something else. If you want the second example in Section 3 to be numbered 3.2 irrespective of how many examples there were in previous sections, then do

```
\newtheorem{eg}{Example}[section]
```

You can use at most one optional argument with each `\newtheorem` command.

You can even number one theorem environment inside another: for example

```
\newtheorem{cor}{Corollary}[thm]
```

if you want the corollaries after Theorem 10 to be Corollary 10.1, Corollary 10.2, etc. Be careful not to create a circle of environments numbered within each other.

#### 6.4 Unnumbered environments

There are several items, such as definitions, remarks and notation, that clearly should be theorem-like environments except that they should not be numbered. It would be possible to set them all up and then separately adjust the counter on each one so that it is not numbered. However, it is easier to take advantage of the number-in-the-same-sequence option. Set up a single unnumbered counter with

```
\newcounter{unnumber}
\renewcommand{\theunnumber}{{}}
```

and then put the other unnumbered theorem environments in the same sequence with

```
\newtheorem{rem}[unnumber]{Remark}
\newtheorem{def}[unnumber]{Definition}
```

#### 6.5 Other systems of numbering

Many Mathematicians want the possibility of having Theorem A, Theorem B etc. as well as Theorem 1 etc. This is no problem. Use the commands given in [4, page 92] or [5, page 98] to alter the way a theorem environment is numbered. Thus

```
\newtheorem{thma}{Theorem}
\renewcommand{\thethma}{\Alph{thma}}
\begin{thma} The subgroups ...
...
\begin{thma} The irreducible ...
\label{char}
...
The result of Theorem~\ref{char} \ldots
```

**Theorem A** *The subgroups ...*

**Theorem B** *The irreducible ...*

The result of Theorem B ...

Other possibilities for numbering are

```
\alph{thma}   Theorem b
\roman{thma}  Theorem ii
\Roman{thma}  Theorem II
```

#### 6.6 Changing the fonts

Fonts are handled differently in L<sup>A</sup>T<sub>E</sub>X 2.09, in NFSS, and in the new standard L<sup>A</sup>T<sub>E</sub>X, L<sup>A</sup>T<sub>E</sub>X 2<sub>ε</sub>. All the suggestions that I give in this section work in both L<sup>A</sup>T<sub>E</sub>X 2.09 and L<sup>A</sup>T<sub>E</sub>X 2<sub>ε</sub>. They do not work at all if you run NFSS without L<sup>A</sup>T<sub>E</sub>X 2<sub>ε</sub>. If you are using L<sup>A</sup>T<sub>E</sub>X 2<sub>ε</sub>, you *must* use the forms like `\sc` given here: the commands like `\textsc` will not do the right thing, because they *add* small capitals (say) to the default fonts instead of *replacing* the default fonts.

The `\newtheorem` command in L<sup>A</sup>T<sub>E</sub>X is the most wonderful thing to happen to Mathematical writers in a long time, because so many of our constructs fit it. However, one of the worst things to happen to Mathematical writers is the hard-wiring of the fonts for the theorem heads and the theorem bodies. Ordinary L<sup>A</sup>T<sub>E</sub>X simply does not give you the flexibility to change these easily. Yet the defaults are not always appropriate, and different journals demand different fonts for these purposes. I suspect that this hard-wiring is one reason that some Mathematicians have been reluctant to use L<sup>A</sup>T<sub>E</sub>X. What can the ordinary user do about this problem?

I shall give four answers, because different solutions are appropriate in different circumstances.

(i) *Bare hands* Sometimes (for example, when sending an article to *Baskerville*) you cannot submit your favourite style files along with your main file. So you need a ‘bare hands’ way of changing the fonts, without losing too much genericity. Here’s how.

To make a theorem environment `prop` whose theorem head is set in small capitals, do

```
\newtheorem{prop}{\sc Proposition}
```

For a small document, this will do. To be more generic, you could do

```
\newcommand{\headfont}{\sc}
\newtheorem{prop}{\headfont Proposition}
```

so that only one line has to be changed if you decide to change the font of all the theorem heads.

To make a theorem environment `qn` whose theorem body font is set in roman, use a two-stage process. The exercises in this sequence of tutorials are defined by

```
\newtheorem{preqn}{Exercise}
\newenvironment{qn}{\preqn\rm}%
{\endpreqn}
```

(The exercises in this particular tutorial are bad examples, because they all have italic instructions.)

(ii) *Mittelbach's style file* Frank Mittelbach wrote the style file `theorem.sty`, which should be available from all good CTAN hosts. It is described in [2, pages 251–255]. It enables you to change the fonts and the layout of theorem environments. However, it does not do exactly what I need.

(iii) *My style file* When I first started to use L<sup>A</sup>T<sub>E</sub>X it was obvious to me that a flexible means of changing the fonts for theorem heads and theorem bodies had to be provided. I hacked `@begintheorem` from `latex.tex` to produce a style file which lets me give a single command to set the font for theorem heads, another to set the font for (most) theorem bodies, and another to say that all subsequently declared theorem environments will have their bodies set in ‘ordinary’ type: not necessarily in roman, but in *whatever font the surrounding text is in*. However, like many others, I deplore the proliferation of personal style files because they inhibit portability of documents, so I haven’t made this style file generally available.

(iv) *American Mathematical Society* The old version of `amstex.sty` (see [1]) gives the user the possibility of declaring theorem environments in three classes—plain theorems, definitions and remarks. However, it does not give the user any control over the fonts used in those environments. I do not know if the forthcoming package `amsthm.sty` will give any more freedom.

## 6.7 Proofs

We all need a proof environment, so everyone invents her own, some more successfully than others. I think that the proof environment should be made with `\newtheorem`, so that all the benefits of spacing, of page-break penalties and of consistent head fonts can be retained. Of course, proofs should be unnumbered and (usually) set in the same font as the surrounding text. So I simply use the foregoing methods to create an environment `pf` with name `PROOF` which is unnumbered and has its body set either in the surrounding text font or in roman.

What should you do about the end-of-proof symbol? Some people want it put in automatically. In principle this could be done with something like

```
\newenvironment{proof}{\pf}%
{\eop\endpf}
```

where `\eop` is your favourite end-of-proof symbol, for example

```
\unskip\protect\nolinebreak\mbox{\quad$\Box$}
```

This is not really satisfactory if you have any proofs that end in displayed Maths (or any other sort of display). Traditionally the end-of-proof sign goes in the display, not on a new line; but if you have a display inside an environment then L<sup>A</sup>T<sub>E</sub>X finishes off the display and gets ready for a new line before it reads the instructions for the end of the environment. So my advice is to have an `\eop` macro and put it in by hand at the end of every proof, either just inside the final display (if this is the last thing in the proof) or just before the `\end{pf}`.

## 6.8 Questions and Exercises

A theorem environment is ideal for questions on exam papers and coursework sheets, and exercises in text books. It is usually better than `enumerate` because it retains the normal textwidth, paragraph indentation and paragraph separation. If you want the questions to be headed simply ‘1’, ‘2’, etc. then do

```
\newtheorem{question}{}

```

If the questions have parts and subparts, it is sensible to use `enumerate` for them. In that case you probably need to change the default numbering of the `enumerate` environments so that, say, parts are labelled ‘(a)’ etc. and subparts ‘(ii)’ etc. The next section shows how to do this.

## Exercises

**Exercise 51** Create a short document with two sections. In the first section put one theorem, a lemma subtitled ‘Burnside’, another theorem, and a remark. The remark should be neither numbered nor in italic. In the second section put another lemma, another theorem, a corollary numbered in the same sequence as the theorems, and finally a theorem in a roman-numbered sequence. Include cross-references to all the numbered items.

**Exercise 52** Redo the previous question, in such a way that lemmas and equations are numbered within sections.

## 7 Other numbered things

### 7.1 Numbered lists

If you use `enumerate` within a theorem environment then you will probably have to change the way that the different levels of enumerated list are numbered. This is controlled by commands containing the strings `enumi`, `enumii`, `enumiii` and `enumiv`. Thus the  $N$ th level of nesting is controlled by `enumN`.

The counter for `enumN` is called simply `enumN`. To alter whether the counter is displayed as an arabic numeral, a letter etc., you change `\theenumN` (see [4, pages 91–92] or [5, pages 97–99].) To alter the printed labels which are put on the items in the list, change `\labelenumN` to be a suitable text containing `\theenumN`.

I find that two levels of nesting are quite sufficient within exam questions and homework problems. My style files for exams and homeworks contain the lines

```
\renewcommand{\theenumi}{\alph{enumi}}
\renewcommand{\labelenumi}{(\theenumi)}
\renewcommand{\theenumii}{\roman{enumii}}
\renewcommand{\labelenumii}{(\theenumii)}
```

In a book, you might need to put something similar in the start of an `exercises` environment.

If you are lazy then you might try to alter just `\labelenumN`. The list items will have the correct printed labels but your printed cross-references will not match. The cross-reference generated by a `\ref` call to a `\label` in the  $N$ th level of nested `enumerates` has the form

```
\p@enumN\theenumN
```

where `\p@enumN` usually picks up the `\theenumM` from higher levels ( $M < N$ ), and possibly some punctuation. If you don't like the settings of `\p@enumN` that L<sup>A</sup>T<sub>E</sub>X gives you by default, you will have to change them in a style file.

### 7.2 Footnotes

Mathematicians usually don't use footnotes, because the footnote marks would be interpreted as superscripts or operators. However, we do sometimes like to put information at the bottom of the first page of a document, under a horizontal line: perhaps an address for correspondence, or a list of AMS subject categories. You can do this with a `\footnote` early in the document, so long as you have first done

```
\renewcommand{\thefootnote}{}
```

It is best to put this command in a small group around the use of `\footnote`.

## Exercises

**Exercise 53** Modify the document in Exercise 51 so that one of the theorems has parts and subparts. The parts should be labelled

[A], [B], ...

and the subparts

1/, 2/, ...

**Exercise 54** Modify the document in Exercise 53 so that the foot of the first page carries the text

Key words: construction of designs; neighbour balance; optimality; randomization; software.

## References

- [1] AMERICAN MATHEMATICAL SOCIETY: *AMS- $\LaTeX$  Version 1.0 User's Guide*, American Mathematical Society, Providence, Rhode Island, (1990).
- [2] GOOSSENS, M., MITTELBACH, F. & SAMARIN, A.: *The  $\LaTeX$  Companion*, Addison-Wesley, Reading, Mass., (1994).
- [3] KNUTH, D. E.: *The  $\TeX$ book*, Addison-Wesley, Reading, Mass., (1984).
- [4] LAMPORT, L.:  *$\LaTeX$ : A Document Preparation System*, first edition, Addison-Wesley, Reading, Mass., (1986).
- [5] LAMPORT, L.:  *$\LaTeX$ : A Document Preparation System*, second edition, Addison-Wesley, Reading, Mass., (1994).
- [6] SWANSON, E.: *Mathematics into Type*, revised edition, American Mathematical Society, Providence, Rhode Island, (1979).

---

## XIII One by one the guests arrive

Kees van der Laan  
cgl@rc.service.rug.nl

---

This note emerged from a request of Sebastian Rahtz triggered by my message which I passed along with my public appraisal for the 100 (L<sup>A</sup>)T<sub>E</sub>X FAQs.

This plea, this shout, hopes to awake the notion that we are all better off if we write macro *software* in the lowest common set of all T<sub>E</sub>X flavours. At least it might initiate a discussion because I'm realistic enough that not all involved share my views :-)).

Of course I know that reality is more complicated, and that a right balance is the best we can opt for, so let us go for that.

### 1 Why,

would a L<sup>A</sup>T<sub>E</sub>X devotee ask? Do you have concrete arguments? Well, from my own experience I can say that there was a time I needed to typeset number ranges. Only the L<sup>A</sup>T<sub>E</sub>X style of Donald Arseneau was available. But, what I needed was a few macros to cooperate with plain, so I had to write one of my own,<sup>1</sup> which by the way emerged as a much, much more compact suite. After all the need has faded away because I tackled the handling of bibliography references more fundamentally. The point is that it would have been better if there had been a kernel independent from the higher layer which I could have taken over. The interface towards the higher level, or let us say the user interface, should better be built on top. The paradigm in this example is the awareness of CISO, as analogy of FIFO, meaning Collective In and Smallest Out, which solves the problem.

That this approach is beneficial in software engineering in general is proven by the various numerical software program libraries, which have the basic material written in the lowest language feasible, FORTRAN, allowing stability, optimization of the code, and confidence in use. Similarly, I remember the PDE (partial differential equation) packages which use common basic algorithms, but differ in the jargon at the user level. I hope that the macro/package/module writers have a feeling for the savings of the costs which can be gained over time, by this attitude. As a volunteer organization one could shrug it off and say I don't care, costs are not relevant. Then there is still another nasty guy lurking around the counter that the (All)T<sub>E</sub>X community like various sects will fall apart, will fragment. To continue the tune

And no one knows where the night is going  
And no one knows why the wine is flowing  
O love, I need you, I need you, I need you  
I need you now

Another example to the point is how to provide for headings? The answer is that I don't care so much about heading macros because the common part is so negligible, while it is highly intertwined with the user interface. But — there is always a but — I for one am strongly in favour of starting from two-part macros, which should perform the essential functionalities whatever you may wish, and build all the ornamentation — i.e., the user-interfaces, eventually with less functionality — on top. This approach obeys the *separations of concerns* principle, and pays off in maintenance, if not that it spreads more easily.<sup>2</sup> To give you an idea of how I did it basically in `blue.tex`:

```
\def\beginhead{<the required functionality>}
\def\endhead{<the required finishing off>}
%with as one-part on top
\def\head#{\bgroup\beginhead
\aftergroup\endhead
\afterassignment\ignorewhitespace
```

---

<sup>1</sup>Who cares? It is estimated that 80% or more of the software is continuously rewritten, that is a fact. My reply is that we can do much better, and we should if we opt for the best.

<sup>2</sup>Forgive me this joke, with L<sup>A</sup>T<sub>E</sub>X widespread.

```
\let\dummy= }  
%or the tribute to manmac  
\def\bluehead#1\par{\beginhead#1\endhead}
```

The last tribute lost the processing on-the-fly functionality, but most of the time I don't need that, at the expense of simpler markup. But the latter is a matter of taste.

If people like a L<sup>A</sup>T<sub>E</sub>X-flavoured header, just go ahead and add it. The fundamental functionalities have been provided already, just a user interface has to be provided as variant.

## 2 Conclusion

The point I'm trying to make is that we are all better off if complex fundamental parts are programmed in 'plain', perhaps after all it has proven to be a fundamental point. To end Cohen's song:

The guests are coming through  
The open-hearted many  
The broken-hearted few

---

## XIV Something is happening, but you don't know what it is

Peter J. Cameron  
School of Mathematical Sciences  
Queen Mary and Westfield College  
Mile End Road  
London E1 4NS

---

This is intended as a worm's-eye view of what is happening to  $\text{T}_{\text{E}}\text{X}$  in the mathematical community at present. It seems to me that there are some problems.

I have earned my living as a mathematician for 24 years. For half that time, the tools of my trade included a portable typewriter and large quantities of Tipp-Ex, and all formulae were written in by hand. For the next six years, I used various word processors; no more Tipp-Ex, but still handwritten formulae. Then I discovered  $\text{T}_{\text{E}}\text{X}$ , and took to it with the inevitability of a love affair. Is it always so difficult for a love affair to last?  $\text{T}_{\text{E}}\text{X}$  and I have started having problems, and perhaps the fault isn't all mine. Maybe we should have some counselling.

$\text{T}_{\text{E}}\text{X}$  fills two roles in the working life of a mathematician, and early on I fell into the trap of confusing them. On one hand, it is for producing masterpieces of the typesetter's art; this was such a delight! For this reason, early on I rejected  $\text{L}_{\text{A}}\text{T}_{\text{E}}\text{X}$ : I was unable to make a  $\text{L}_{\text{A}}\text{T}_{\text{E}}\text{X}$  document look good; and while ordinary mortals can write or edit  $\text{T}_{\text{E}}\text{X}$  macros, only superheroes can mess with  $\text{L}_{\text{A}}\text{T}_{\text{E}}\text{X}$  style files. In plain  $\text{T}_{\text{E}}\text{X}$ , with the help of invented or adapted macros and a variety of fonts, I made everything from books, through class exercise sheets, to character sheets for my son's role-playing games, all stamped with my own design (for better or worse).

The other aspect of  $\text{T}_{\text{E}}\text{X}$  relevant to mathematics is its function as a communication standard. Several things contributed to this, for most of which we have Knuth to thank. Most important is its free availability on all platforms, and the fact that the input is ASCII text without control characters, so that it can be sent by email without damage (even to and from the UK nowadays, although it wasn't always so!). Gone are the days when international collaboration involved exchange of letters taking weeks, with the inevitable crossing of information. I can now exchange drafts of papers with co-authors almost instantaneously (though, of course, time differences mean that I usually have to wait a day for a reply from Australia).

Another significant fact is that email and  $\text{T}_{\text{E}}\text{X}$  provide a mechanism for communicating formulae in an email letter. Between  $\text{T}_{\text{E}}\text{X}$ -literate correspondents, such snippets of  $\text{T}_{\text{E}}\text{X}$  are never compiled (except, arguably, in biological computers).

Besides the two mentioned above, a third factor has contributed to establishing this standard. Knuth, as a mathematician, designed the plain macros to be as close as possible to the way that formulae are pronounced by mathematicians. Thus,  $x \over y$  produces  $\frac{x}{y}$ . (One of the few exceptions to this is that we have to say  $\bar{x}$  rather than  $x \bar{\phantom{x}}$  to get  $\bar{x}$ .)

My first encounter with  $\text{T}_{\text{E}}\text{X}$ , before I knew what it was, came about when an editor sent me a referee's report in uncompiled form. The dollar signs were a bit mysterious, but if I ignored them, the rest made quite good sense!

Regrettably, many popular macro packages have lost sight of this point, and seem obsessed with the need for all operators to be prefix. Mathematicians, brought up with the infix and postfix arithmetic operations, are free from this artificial hang-up, and never refer to  $\frac{x}{y}$  as  $\text{\frac{x}{y}}$ . (Well, maybe not quite — but this is certainly true for formulae simple enough to be put into a letter.)

Now there is a clear conflict between these two roles of  $\text{T}_{\text{E}}\text{X}$ . It was borne in on me when I wanted to send my beautifully-crafted preprints to other people. At best, I could send along several macro files, and assume that my correspondents could follow the instructions for naming and using them and cope with the organisational problem. At worst, the recipient would lack a font I'd used, and would be unable to print the document at all. So, inevitably, I was forced into keeping two copies of each file, a fancy one for myself, and a plain one for everyone else. This made updating the files a nightmare, especially when one was at home and the other at work. The next stage was to abandon the fancy files, and keep everything as plain as possible!

These problems, if understandable, were at least self-inflicted. But it seems to me that the academic/publishing community is now falling into the same trap.

It is now very common for publishers to encourage electronic submission of manuscripts. Among the specialist journals in my field, with one rogue exception which specifies WordPerfect (stop laughing at the back!), the system of choice is  $\LaTeX$ , with a proprietary style file to reproduce the existing look of the journal. Some of these style files are less than perfect. (One publisher, attempting a clever redefinition of `\emptyset`, ended up leaving this command undefined. Another insists on printing the journal's copyright message on my preprints.)

Often, these style files tempt the author with added features, from the trivial (an `\email` command to print the author's email address) to the valuable (a `proof` environment for the proofs of theorems). If you bite the apple, you can no longer compile your paper in ordinary  $\LaTeX$ , and so you can no longer email it to your collaborators. Yet some of the features are too good to miss, and the journal makes others compulsory. So, once again, I have to maintain two copies of my files.

Further problems are caused by the proliferation of  $\LaTeX$  versions and font selection schemes. Rather than stick to the lowest common denominator, some journals provide elaborate format-switching mechanisms whose instructions are very difficult to decode.

Electronic journals pose still more problems. We are told that this is the way of the future, and that traditional journals will quickly die out. Yet I am sure that many academics, (and not only in the Third World), are unable to read or access these journals. We get busier and busier as time goes on, and installing Mosaic and all the necessary supporting software on your computer is a non-trivial job. And on a more mundane level: an otherwise excellent electronic journal in my field has, as virtually the only style specification, the use of `cmcc8` for the headline. This font is not in the  $\emTeX$  distribution. Fortunately, the `.tfm` and `.pk` files for this font were available on our Unix machines, so I was able to download them and correct the lack. How many beginning net surfers would be deterred, by choice or necessity, by just such a small irritation?

If publishers do force us into using discordant versions of  $\TeX$  by such means, then the role of  $\TeX$  as the standard for mathematical communication will be threatened. If this is lost, one of the major advantages of  $\TeX$  over other systems will go with it. Can we save the situation? I do not believe that standards can be established by wishing for them, even by formalising the wishing into a committee. The only thing that seems to work is the commercial success of particular hardware or software. But what could we ask of  $\LaTeX3$  (or whatever is to be the standard)? Just two pleas come from the concerns I have raised:

- It must be possible to impose different styles with the absolute minimum of change to the input file. This means that all publishers' requirements must be anticipated and default versions included in the standard style. Sounds totally impractical? But we know what happens if you don't!
- Either all the plain  $\TeX$  mathematical commands should be available, or the commands that are actually used should conform as closely as possible to spoken mathematics.

---

## XV Malcolm's Gleanings

Malcolm Clark

---

### 1 Spivaking anyone?

Cybernauts will be familiar with LambdaMOO, one of the information superhighway's more recherche laybys. For those with both feet in reality, a MOO is an object oriented MUD, and a MUD used to be a multi-user dungeon, but has achieved respectability by becoming a dimension or discussion (depending on whose acronym cracker you use). When it was a MUD it was just an on-line game for propellor heads (usually male, usually adolescent), without the benefit of graphics. Just a text based dungeon and dragons game. The sort of thing your average T<sub>E</sub>X head would enjoy. In its new incarnation it has become a useful conferencing tool (as well as a virtual world for role players). Xerox PARC (the guys who brought you the first usable graphic user interface while the Steves, Jobs and Wozniak, were still cadging chips from Mr Hewlett and Mr Packard) is the home of the LambdaMOO, where you can register to have your own room which you organise as you wish and to which you may invite whomever, or maybe even whatever, you choose. A plausible version of what this might become is contained in Neal Stephenson's cyberpunk novel 'Snow Crash'. What has this to do with the world of T<sub>E</sub>X, apart from its similar single-mindedness? Just that 'to spivak' is a way of describing one sort of role-playing. More research needs to be done to establish the full implication of this. A prize for the first entertaining (if inaccurate) definition: LambdaMOO may be found at the URL: `telnet://lambda.parc.xerox.com:8888` (that's enough to separate the kids from the lambs).

### 2 Stability or statis

The latest round of 'corrections' to the T<sub>E</sub>X suite has just been released by Donald Knuth. It includes adjustments to T<sub>E</sub>X, Metafont and the Computer Modern fonts. The bumper cheques (top amount this time, \$327.68) went to the legendary (if elusive) Chris Thompson and Bogusław Jackowski. This takes T<sub>E</sub>X to version 3.14159, and Metafont to 2.718. The announcement was accompanied by the statement that the next and successive rounds would occur in February "1998, 2002, 2007, etc!". In line with all best laid plans, no sooner had the toner on my laser printer fused than another bug was found and corrected. The numbers remain the same though.

### 3 TUGboat

*TUGboat* has arrived. At least, volume 15, number 3, the conference edition has made it to our shores (coincidentally at the same time as it turned up in Santa Barbara). It's a reasonably thick compendium, despite omitting a few of the papers which were presented. The omissions are either because the articles were very similar to already published material (like Rowley & Mittelbach, and Bigelow), or because it will appear in a future issue (like Hosek, Haralambous, and Laugier & Haralambous), or, rather oddly and without explanation, withdrawl (Haralambous). It's better and more cohesive than I remember at the conference. The major innovation is the inclusion of several pages in colour, appropriate at a meeting where so much attention was directed at the use of colour. In contrast to the edition of Cahiers GUTenberg which used colour integrated with the text, all the colour examples are included in an Appendix. Many of the new extensions to L<sup>A</sup>T<sub>E</sub>X reflect or anticipate the widespread adoption of colour. This volume may be a timely summary of many of the issues and consequences. But there is much else there.

### 4 A few last words

I continue to be surprised by the attention that this column attracts. In my view it is a filler which helps the editor to pad out a few columns and the only balance it achieves is purely in those column terms. It is not to be taken seriously.