Formatting SGML Manuscripts*

Jonathan Fine

203 Coldhams Lane,
Cambridge CB1 3HY, England
j.fine@pmms.cam.ac.uk

1 Formattting SGML Manuscripts

This article is about typography, SGML, TEX, and SIMSIM, which is a new TEX 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 WYSYWIG 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.

2 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 TEXbook*, 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. TEX'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 LATEX) 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:

3 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.

*Reprint from the Annals of the UK TEX Users Group **Baskerville**, Volume 5.2, March 1995. Published with permission of both Baskerville editor and author. Presented at the UK TEX Users Group conference 'Portable Documents: Acrobat, SGML, and TEX', on 19 January 1995, London, England.

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 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 occassion to impose their will on the software's production of the formatted document.

4 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. 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.

5 What is TEX the program?

TEX 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 TEX one is given a modest monetary reward for finding a bug.

It is worth remembering that LATEX is not only a macro package but also an input file syntax. Because TEX is programmable, no fixed input syntax is required. Given sufficiently tricky macros, the mighty lion that is TEX 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 TEX approach, when confronted with SGML files to typeset, is to translate into LATEX or the like before calling on TEX 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 LATEX has with verbatim in titles and so forth, and the limitations should become apparent. Translation to LATEX might have been the best there was available, but it is certainly not the best that is possible.)

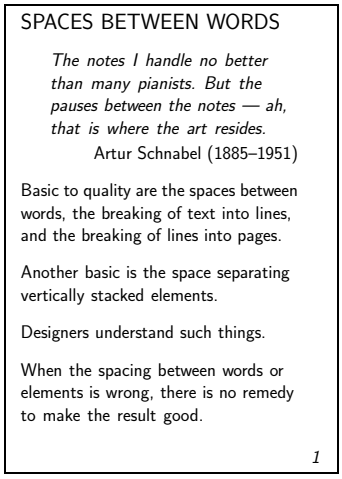
6 What is simsim?

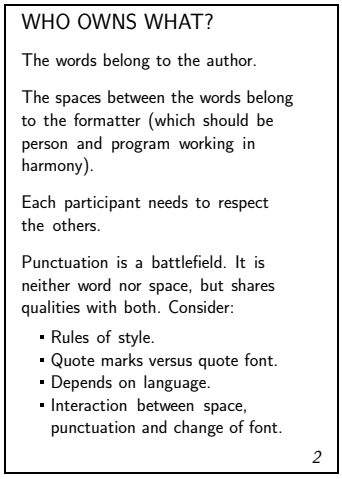
This brings us to the final part of the talk, which is a software announcement. The OHPs were typeset using a preliminary version of a TEX macro package SIMSIM that I have been developing for several years, and which is close to completion. The English word 'sesame' is already a registered computer software 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.

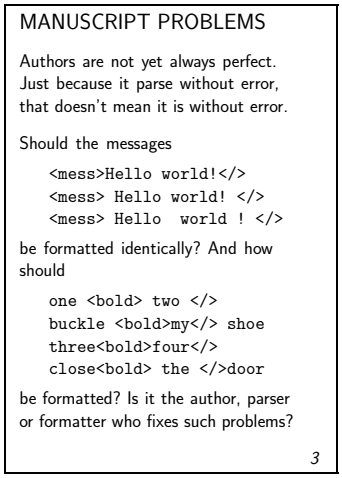
There are two sides to SIMSIM. Input and output. Input is SGML and also style files. Output is pages formatted by TEX. The title slide of the talk was typeset from:

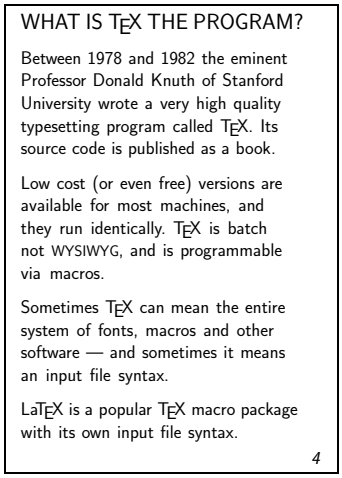
```
<title-page title =
"FORMATTING / &SGML / MANUSCRIPTS /
 - or - /
MUCH ADO / ABOUT /NOTHING"
>

<par> UKTUG and BCS-EPSG meeting </>

<ol>
<li> (c) Copyright 1995 </>
<li> Jonathan Fine </>
<li> 203 Coldhams Lane </>
<li> Cambridge </>
<li> CB1 3HY </>
</ol>

</title-page>
```

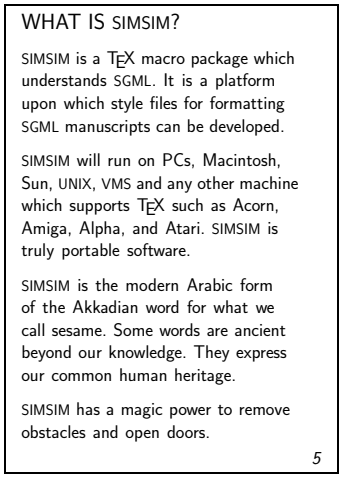
Notice that the title has been entered an attribute value, with the line breaks denoted by forward slash '/' or solidus characters. This is a notation in wide use for displaying

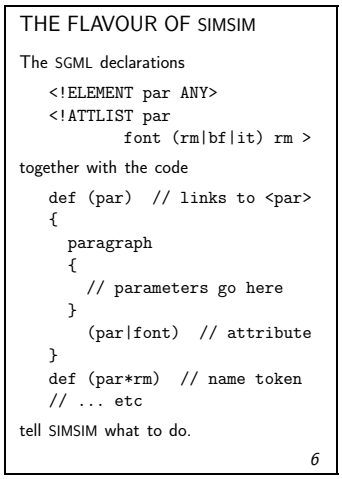


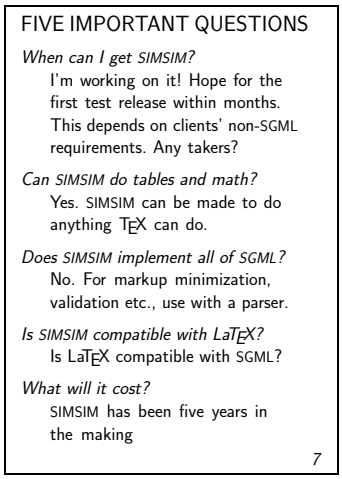












line breaks in verse quoted as flowing text within a paragraph. Suppose one were presented with the title slide and were asked to encode as an SGML element. This is the sort of thing that the Text Encoding Initiative Guidelines were developed for. One would record that it was a title page, that such 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 refence. 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.

7 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 appropiate 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` entitities.

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.

8 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 `<bl>` rather that `<ol>` list. This is because the action attached to a tag is held as a TEX control sequence token, whose meaning can be changed just like any other control sequence. So the token represented by `(bl)` 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 LATEX? So far as I am concerned there is no relation with LATEX, and no means of converting documents from one form to another. Or style files for that matter. SIMSIM and LATEX both start with uninitialised TEX, but from there proceed in different directions and with different assumptions. I don't see any interaction between the SIMSIM and the LATEX 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 TEX 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 LATEX. 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 TEX)? 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.