TUGboat BLUes— how TEXies do it[*]

Kees van der Laan

Abstract

The significance of *TUGboat* for the TEX community at large is praised.
(l)tugboat.sty, tugboat.cmn and the TUG authors' guide are discussed, next to their siblings (l)tugproc.sty and the option eurotex.sty.
Article templates for the various 'styles as is' are provided. Independent and in addition to these I included customing files. Also included is my concrete proposal for a tug.ppt style—for preprints of tugboat.sty, and tugproc.sty, in the spirit of ams.ppt.
Furthermore, a new and simple alternative to the handling of options for TEX is proposed, based on the toks variable *this<foo>*, analogous to *every<foo>*. This is applied to handling verbatims, with as a result a compact suite of verbatim macros to be used with AnyTEX.

Keywords: Block comment, computer-assisted typography, education, eurotex.sty (option), (LA)TEX, macro writing, mark up, optional parameters, preprint style, publisher formats, TTN, tugboat.cmn, (l)tugboat.sty, (l)tugproc.sty, two↔one-column format, verbatims.

1 General

1.1 Introduction

This is the third paper in the series about general, public domain macro collections to mark up and format (typographically) complex math-oriented documents.[1] This delving into literature has all to do with my 'don't rush into code' attitude. I like to study the past and build upon that.

Reality has it that *TUGboat* has functioned as the state-horse of the TEX-community, and stimulated many a TEXie. It has also been the example of how a TEX bulletin should look like for the various LUGs. Whenever the question arose what TUG has to offer its members, the answer has invariably been *TUGboat*. This limited perception has been counteracted since 1990 or so, via the flyer[2]

> '8 great reasons to join TUG'

Moreover, we have now also the portable TTN—TEX and TUG News. It is perfectly clear what TUG has to offer its members. Real leadership! This paper is aimed at a broad audience

- LUG editorial boards, who like to become informed about the TUG styles
- TUG editors/producers, who welcome feedback
- TEX hackers, who might learn a lot from the `tugboat.sty` codings and the proposed alternatives
- and, in general those who like to hear what I have to say.

The paper is not aimed at those who just jump into it for search of typos or typographically non-optimal aspects, and who are not really interested in the results of my research.

The TUG styles—and their descendants—represent different points of view. The total of the material treated in this article has therefore the potential of being very confusing, mixing up all those different ideas about macro writing and setup of style files in the mind of the reader. In order to circumvent this to happen I would suggest readers to glance through the introductory material up to the 'tugboat.sty' section, and to skip my ample footnotes and intermezzos on first reading. Then—depending on your TEX roots—peruse either the parts

- `tugboat.sty`, and `tugproc.sty` with their corresponding templates as provided in the Appendices for plain TEX trusties, or
- `ltugboat.sty`, `ltugproc.sty`, `eurotex.sty`, and `ttnxnx` with their corresponding templates as provided in the Appendices for LATEX devotees.

And for those who just like to lay hands on the new suite of verbatim macros, jump into 'Alternative: the big deal,' in the verbatim section.[3]

An inherent weakness of these kinds of papers is that the material treated is dated.[4] Being aware of this I'll not only concentrate on telling what is in there, but I'll also discuss it, question it, and show alternatives. In short I'll go for

[*] Parting gift to the BoD of TUG on the occasion of me leaving this board as a special director.
[1] The preceeding papers in the series are Manmac BLUes and AMS BLUes. BLU stands for Ben Lee User of the TEXbook fame. BLUe is its cousin which I adopted.
[2] Contact the TUG office: tug@tug.org.
[3] However, in BLUe's Verbatim I provided a revised suite.
[4] Mind the version numbers of the styles!

providing insight.[5] When other versions come around it is hoped that the reader can better appreciate these—at least I can now I guess—because he/she knows what is going on, what the pros-and-cons are. In short the lasting value should be that the reader has built a 'hat-rack' to appreciate new—or related—versions, the 'hats of TEX.'

There seems to hang a sort of taboo—we are the champions, aren't we?—around these kinds of papers. Invariantly organizations perceive them as an attack on their 'precious child.' Let me state it loud and clear: it is not an attack, and definitely not meant as such. If I misperceived anything—unintended :-) of course—it's my fault and that has nothing to do with the subject treated or the responsibility of those involved in the design of the style files or `TUG<whatever>` production.

With respect to the templates I distinguished two issues. First, templates are provided for the various articles, with the use—*as is*—of the `<style>.sty`-s. Second, there are the customing files, to allow for page sizes, running headers and footers of your own. I rejected the idea of splitting up the article into a bunch of smaller ones dedicated to for example TEX and LATEX, respectively. In my honest opinion (MHO for short) the total is the most interesting, to have it all together.

To be honest this article is mainly a stimulus for me to

- better understand the *TUGboat* styles
- give feedback to those involved in the *TUGboat* et cetera production, and to those who take care of bringing out the proceedings of the various TEX meetings
- have all the templates and customings together.

This paper is also a parting gift to the Board of Directors of TUG, because of my resignation in June 1994 as NTG's president I won't be a special director of the BoD of TUG any longer.

Personally, I consider my new alternative code for the handling of options via `\this<foo>`, which is analogous to `\every<foo>`, very simple and useful.[6] See the verbatim section in the part '`tugboat.sty`.'

Although this article is critical about 'How TEXies do it' it is tacitly meant as a positive contribution. It can assist the selfinspection process. Be happy with the good news, and be realistic about the *inelegant* ways we mark up, or write TEX code now and then. Dare to question the a priori choices and coding conventions, adopted a decade or so ago. Re-ponder the effect of wishful thinking especially in a volunteer-based world. Realize, accept and discuss! This will strengthen our efforts to provide for the best in a multi-dimensional sense.

1.1.1 Warnings!

The paper is incomplete, and certainly not the last word on the issue. It is not easy-reading either. Some 15 years of experience have been glanced through, and the area surveyed is broad. Add to that some new ideas, and that TEX is inherent complex, subtle, or at least unusual. Moreover, related areas such as literate programming and hypertext, are touched upon. As a result nobody would be really surprised to find that a paper like this is hard to read, when details are also dealt with. At the very least a reader should take his/her time for reading it, and make a selection to start with. It is dated too. Some elements have not been treated equally in all parts, probably because I considered them not that relevant, for the moment. The macro code discussions are different from just inclusion of macro texts, followed by examples of their use. Therefore, I can't possibly guarantee that they are all correct. However, I have done what I could. I have separately tested the verbatim suite and the templates at the end. Furthermore, my aim has been to convey insight not to provide production alternatives.

1.1.2 Generic coding pitfall

To start with I will discuss a pitfall of mark up coding I tumbled in myself.[7] In order to have a generic footnote command which automatically keeps track of the footnote counter—to be customized to plain (or LATEX)—I provided

```
\def\ftn#1{\advance\fcnt1 %counter details
  \footnote{${}^{\the\fcnt}$%..typesetting
   }{#1}}                   \newcount\fcnt
```

See what happened with respect to plain's footnote?

The footnote text is no longer processed on the fly!?!

I should have omitted the argument—mea culpa— just a parameterless shortcut. TEX is so unusual! Deteriorations like these are easily coded and innocently passed on, which is even worse.[8]

1.1.3 Why?

There are a few reasons why it is worthwhile to study *TUGboat*'s styles. The styles

- embody some fifteen years of experience
- are used in practice
- evolved, driven by the needs of advanced TEXies
- facilitate submissions marked up in the TEX or LATEX spirit

[5] Although, to paraphrase Hamming: The purpose of literate programming is not yet in sight.

[6] Comparable in functionality to Knuth&Levy's verbatim mode.

[7] In good company, though. LATEX's footnote suffered from the same pitfall. (It is also in `\@makefntext` in the `ltugboat.sty` section.) I did not stumble on things like this in `tugboat.sty`. It is mentioned at the beginning of this article to illustrate the subtleness of macro writing in TEX. TEX is so unusual.

[8] BLU might say 'Who cares?' Well, I do and *you should too,* because in the wrong case you can't mark up verbatims in footnotes as you are used to. Nobody—certainly not flexible and tolerant TEX—will complain. To be concrete verbatims like `|<special character(s)>|` will go wrong, in the sense of unexpected results. (The use of the alias tag '|' is irrelevant.) A further enhancement is to use a `\global` advance of `\fcnt`. Furthermore, this example is fundamental. It stimulated me to change my btable macro into a two-part macro, such that verbatims—read special characters—can be processed too. I also appreciate Knuth's `\beginchart` and `\endchart`—from manmac—much better now.

- allow switching from one- into two-column format and vice versa[9]
- allow inclusion of files in verbatim (This enhances consistency of the macros used and those listed. Obligatory for authors who publish about (LA)TEX in (LA)TEX.)
- have implemented a general and orthogonal optional parameter mechanism, and allow for short command delimiter forms
- `tugboat.sty` shows how to handle arguments as stored information and more importantly how to process 'arguments on the fly'
- some macros/utilities can be used in other contexts, for example abbreviations and logos (this is trivial)
- `ltugboat.sty` is an example of how to customize LATEX's `article.sty`.

1.1.4 What?

First I'll pay attention to the provided functionalities and answer the question whether it is easy to switch between the styles. Then I'll discuss how the results in print look like, how the production process goes, what is provided by the style files, what I know of the design, and finally I'll distill most of the coding conventions. Next the `<style>` files are discussed each as a separate entity. At the end I'll also discuss *TUGboat*'s descendants

- `(l)tugproc.sty:` the styles used for the proceedings of the TUG annual meetings
- `eurotex.sty` the option used for the EuroTEX meetings
- and, en-passant touch upon TTN's style.

Each style has a users' guide, of which the article of Beeton and Whitney is the most widespread. Barbara Beeton has surveyed the production history in her '*TUGboat* production: TEX, LATEX, and paste-up.' Copy can be submitted with

> TEX- or LATEX-oriented markup.

In the 'Appendix: Templates' I provided a sequence of template articles—the `<style>.tem` files—which show the essential markups. This anthology makes it easy to compare the various markups. One can't get around the inconsistencies, alas. In the 'Appendix: Customing' I provided the `<style>.cus` files, for modifying the page size and the running headers and footers. The 'Appendix `tug.ppt`' contains my proposal for a preprint style for TUG (and LUGs). The last appendix contains the table of contents.

Functionalities. In a publication I need for markup

- at the outer level: title part, keywords, abstract, headings, floats, footnotes, margin notes, appendices, bibliography, and index
- at the inner level: fonts (coupling and selection), special paragraphs (quote, lists), (file) verbatim listings, programs, pictures (graphics), tables, display math, automatic numbering, (symbolic) cross-referencing, and citations.

Functionalities: (l)tugboat.sty

	`tugboat.sty`	`ltugboat.sty`
title part	++	++
keywords	plain	+
abstract	plain	+
headings	++	++
floats	plain	++
footnotes	+	+
margin notes	discouraged	LATEX
appendices	plain	+
bibliography	plain	+
....	....	....
font	plain	NFSS
quote	+	+
lists	++	+
verbatim	++	++
file verbatim	++	+
escape verbatim	++	+
programs	none/plain	none
pictures	none/plain	+
tables	plain	LATEX
display math	plain	LATEX
aut. numbering	none	LATEX
(sym) cross refs	none	LATEX
col switching	in progress	LATEX
citations	none	LATEX

++ excellent + good

This thinking at two levels is fundamental, although the borderline is ill-defined. In the accompanying table I have enumerated these functionalities and indicated roughly to what extend they are supported by `tugboat.sty`, respectively `ltugboat.sty`.

From the table it appears that the question

> Whether switching from `tugboat.sty` into `ltugboat.sty` or vice versa is easy?

is ill-posed, because the styles support different functionalities, of yet.[10]

The prime developers are Ron Whitney and Barbara Beeton for `tugboat.sty/cmn`, and Adrian Clark, Frank Mittelbach, and Rainer Schöpf, for `ltugboat.sty`, all coordinated by Barbara Beeton, the editor. The TUG styles

[9] Not provided in full generality of yet—in test phase though—that is, it is not possible to switch within an arbitrary page. `tugboat.sty` allows one-column at the first page, and LATEX—and thus `ltugboat.sty`—clears the page after a switch. Two-columns is not supported by the `amsppt.sty` style. In manmac it is used only for formatting the index (no floats in there!). For LATEX Frank Mittelbach has released `multicol.sty`, and I know of some adaptions for plain.

[10] Nevertheless, I make it a habit to mark up headings via `\head*...*` et cetera and to precede in-line verbatims by `\verb`. (The latter when working with TEX redefined as empty.) Similarly, I use `\small` and `\tiny`, next to `\thepage`. The footnotes are marked up by `\ftn` to be customized to plain or LATEX, given the context. For the markup of control commands I use `\cs`. A first step towards generic—context-independent—markup.

are owned by the TeX Users Group, and released in the public domain.

1.1.5 Notations and definitions.

`\ea`, `\nx`, `\aa` and `\ag`, are used as shortcut notations for `\expandafter`, `\noexpand`, `\afterassignment`,[11] and `\aftergroup`, respectively. TUG denotes the TeX Users Group, and LUG stands for Language or local Users Group. Text borrowed from the users' guides and the style files is surrounded by single quotes. (I never use double quotes.)

!?! denotes that I'm much surprised. ?!? denotes that I can't believe my eyes.

Orthogonal facilities mean independent facilities which can be used as such, but when combined will yield new possibilities, without extra coding.

With respect to the use of fonts I'm a bit loose. When I denote *TUGboat* it might mean the journal, the editors, the readership, ..., depending on the context.

1.2 What does TUGboat look like?

First of all it looks great! More seriously *TUGboat* is two-column biased. With respect to the contents major parts are: general delivery, philology, fonts, graphics, book reviews, typesetting on PCs, tutorials, letters, macros, abstracts, news and announcements, late breaking news, TUG business, forms, advertisements. This list reflects the outer level of the table of contents, as supplied on the back. (This classification is distilled from a *TUGboat* issue, and it might be that there is no rigid classification decided upon.) The following issues are also distilled from the copies I own.

Major parts start with the title framed. Each contribution within a part is separated with a `\hrule` from the previous one.

From the outer level point of view each article has the usual structure
- title part (title and author name)
- copy proper (abstract, sections, acknowledgements)
- back matter (references, appendices, signature (name and address)).

There is no uniformity with respect to section numbering and punctuation of the titles.[12] The ordering of the items vary in the back part, for example, the signature is not always at the end. Not every article has an abstract[13] or a keywords part. References are set differently by authors. Labeling and citations vary. As floating objects (insertions) we have page-wide tables which appear at the top of a page or at the end of the article, next to small one-column tables. Footnotes are column-wide.

Remark. Of course there should be 'exceptions to the rule,' especially when an article is about a special formatting issue and the result is demonstrated by the article itself. As an example one could think of Mittelbach's multi-column article. Also some authors have insisted upon one-column format throughout.

1.3 TUGboat processing

The TeXnical production of *TUGboat* can be summarized[14] by the following streams

$$\begin{tikzcd}
\text{author(TeX)} \arrow[d] & \text{author(LaTeX)} \arrow[d] \\
\texttt{tugboat.sty} \arrow[d] & \texttt{ltugboat.sty} \arrow[d] \\
\texttt{tugboat.cmn} \arrow[d] & \texttt{tugboat.cmn} \arrow[d] \\
\text{TeX} & \text{LaTeX}
\end{tikzcd}$$

Next to this there is a refereeing system (from 1990 or so onward). Multiplication of the blues is done by a printing house. The kernel of the editorial board has been proven to be stable and has been in charge for many years.

An alternative approach is there a reasonable one? Let us not bother about the practicalities and concentrate first on how it could have been done best, TeXnically. An alternative is to allow for a LaTeX author interface next to the TeX one—with common lower level markup—as depicted by the following scheme

$$\begin{tikzcd}
\text{author(TeX)} \arrow[d] & & \text{author(LaTeX)} \arrow[d] \\
\texttt{tugboat.sty} \arrow[r] & {} \arrow[d] & \texttt{ltugboat.sty} \arrow[l] \\
& \texttt{tugboat.cmn} \arrow[d] & \\
& \text{TeX} &
\end{tikzcd}$$

Superficially this looks the same but it is not, because there is no LaTeX in there. The *TUGboat* look-and-feel in print

[11] Yes I know, the control sequence is already in use for the Ångstrøm unit, or to denote the å character of the Scandinavian alphabet. I don't use those in here. Just a mnemonics shortcut that is all.

[12] The use of `\nopunctuation` suppresses the end dot after a subhead title. Then none of the heads gets automatically an end dot, not even the running-in subheads. This is simple to remember. From a functionality point of view there should be no default dot. It must be left to the authors whether they need, for example, questions (with question marks) as titles of a heading.

[13] At the end of the users' guide it is recommended to provide for abstracts. (Abstracts are provided in German at the end of each *TUGboat* issue, from 1990 or so onward.) It is much clearer to have it included in the 'outer form' list and incorporated in the template to come. Barbara Beeton communicated that one of her wishlist items is to provide for Author Guides, similar to those of AMS. Volunteers go for it!

[14] Note that this is an oversimplification, it restricts attention to TeX and LaTeX, and abstracts from METAFONT interaction, the inclusion of PostScript, and the like. As I have heard on many an occasion '*TUGboat* production is not simple.' There is nothing against it to try for making it simpler, isn't? In MHO, and with all respect, there is no other way.

is inherently guaranteed, and the LaTeX user can use the commands he is familiar with.[15]

> However, this idea is not viable because the styles differ significantly by the provided funtionalities. Moreover, it is no sinecure to reimplement the LaTeX features in `tugboat.sty`, or the other way round. Furthermore, the temptation will invariantly be to do it in the best way thinkable, and that for sure will take time, as we can witness from the LaTeX3 project. However, some if not all, has been done by Spivak in his LAMS-TeX. Hmmm, puzzling.

Another idea is to start from a generic style which can be coupled to `tugboat.sty`, LaTeX, or you name it. The latter approach is interesting for authors because *their* interface remains the same.[16] De facto LaTeX functions more or less this way. That is

> LaTeX $\equiv$ the *de facto* user interface.

The point I like to get across is that it would be nice when TeX *and* LaTeX marked up submissions can be processed with the 'markup in the small' by TeX. For example for math and tables the macros from plain can be used.[17] LaTeX's variants for these are syntactic sugar, in MHO with all respect. No extra facilities have been added there. LaTeX's greatest contribution is the users' guide aspect in contrast with the TeXbook which contains it all. Lamport '*dared to do less.*'

An annoyance is to have to remember that the markup of a reference to a *TUGboat* article needs awareness of whether I'm using `tugboat.sty` or `ltugboat.sty`. This because of the following borrowed from `tugboat.cmn`.

```
\ifx\tugstyloaded@\plaintubstyle
   \def\tubissues#1(#2){\TUB~#1, no.~#2}
\else
   \def\tubissues#1#2{\TUB~#1, no.~#2}
\fi
```

Why? There is no reason for 'upward compatibility.' It sins against the generic idea. In MHO—with all respect—these kinds of low level

> markup should and *must* be independent from the context.

There are more such context dependencies, as mentioned in an earlier footnote. Other discrepancies come from `tugboat.sty`'s `\article` versus LaTeX's `\maketitle`. Why not redefine `\@maketitle` and supply this redefinition in `tugboat.cmn`?

Agreed, for most authors it does not hinder. They follow one of the streams.

My rule for issues like these is: When in doubt to enforce one way or the other on your user, *don't!*[18]

Refereeing is in MHO—with all respect—a bit misplaced and takes too much energy. Roughly the same results can be obtained when submissions are accompanied by a recommendation of a friendly colleague. This approach is more in agreement with the character of a user group: to assist each other, instead of rejecting—rightly or wrongly—a contribution. And this is much more efficient too. Furthermore, who referees the referees? And what for objective selection process is there to become one? I blame the current process from experience, where despite the guidelines for referees the referee reports vary enormously, not in the least in structure, items treated and tone.[19] I guess the guidelines are insufficient and should pay more attention to the right, respectful and helpful *attitude* of the referees. It remains a difficult and delicate matter. I know of the wishful arguments that the journal status is sought, to benefit from belonging to the class of 'refereed journals.' Perhaps, I'm a bit naive but the quality of a journal is judged by the peers in a field. In the (small) area of a specific computer language—and TeX can be seen as such a language, but admitted wholeheartedly a brilliant one—I have not heard of a 'real journal,' and don't believe *TUGboat* will become one, simply because we need a broader umbrella, for example a journal about electronic publishing like EP-ODD. Don't suboptimize!

It may look like that I'm treated badly—there is some truth in there ;-) —but that is not the point. What I like to get across is that it has all to do with

> get your priorities right and delegate

especially when there is still so much to do for 'finishing' and polishing the styles. Furthermore, it is a volunteer-base world, meaning we are always short of volunteers, and invariantly suffer from lack of time.

1.4 Design

The *TUGboat* styles have not been 'designed.' That is: to my knowledge there is no description of how the journal should look like in its total, the specs so to say to which the style files had to conform. It started with 'a package based only on `plain`,' and[20]

[15] So the question arises whether it is feasible to start from a set of specs (very close to `tugboat.sty`, I presume), 'implement' these as `tugboat.sty`, and provide a LaTeX user interface on top. From the maintenance point of view the answer should be yes. From the human factor point of view the answer is apparently no. The financial means are modest too. I pity that. For a finger exercise of the approach see the section 'Lists,' in the `tugboat.sty` part, for variants with respect to LaTeX's `\itemize`, based on Knuth's `\item`.

[16] A disadvantage is, however, that the appearance will vary with the coupled style. When we look at it from the preprint point of view then this disadvantage turns into an advantage. We want then different appearances in print!

[17] Let us assume an appropriate authors' guide exists.

[18] On second thoughts one should not enforce at all, always leave the choice with the user.

[19] Note that I did not say anything about *what* is stated in a referee report. Too many opinions on that issue, and that is the referee's freedom. But first the form things, the trivia. And if that is settled we can go for the quality aspects of the contents. When the form aspects are not sufficiently settled it is of no use going further.

[20] A lot of thought has been given to consistent coding of the various tags! Top-class, but perhaps too smart.

'...later, as demand for style files follows wherever LaTeX-devotees wander...'

a *TUGboat* variant of LaTeX's `article.sty` was also created. This has been called `ltugboat.sty`. Common matters have been split off into `tugboat.cmn`.

The users' guide mentions euphemistically[21]

'The two macro sets yield much the same output, differing in certain ways for input.'

No results have emerged in public so far with respect to the markup of bibliographies, pictures, symbolic referencing or citations other than LaTeX's approach.[22] The recent practice of handling encapsulated PostScript has not been included in the users' guide of yet. The latter is used in practice by some authors already, as was communicated to me by Barbara Beeton.

To end up this section I still believe that

specifications are mandatory,

and embody 'half of the work,' if not more. Furthermore, I would write nowadays style files in the form of literate programs, as suggested by the polish GUST at Aston, based on specs. Furthermore, by this approach as communicated by Włodek Bzyl the descendants can be implemented via change files. That is all. Neat isn't it?

1.5 TeX coding conventions

For `tugboat.sty` a strict uniform and orthogonal TeX coding philosophy has been adhered too. For `ltugboat.sty` LaTeX's `article.sty` has been used and ipso facto its coding style. In view of the LaTeX3 project we might expect a big change with respect to the latter.

1.6 What is provided by the styles?

From the authors' guide for *TUGboat*—which by the way is a nice paper in itself—we see that the following functionalities have been provided for

- at the outer level: title, author, address, article proper (section structuring with paragraphs (quotes), lists, verbatims, floating figures), and signature
- at the inner level: the details of display math, tables and figures are not described. One is referred to TeX (respectively LaTeX) or specific macros.

Tags for keywords, abstract and bibliography were introduced along with LaTeX.

1.6.1 Contents of the style files

`tugboat.cmn` contains among others the formatting of abbreviations like `\AMS` for American Mathematical Society, `\aw` and so on. Very handy for enhancing consistency.[23] Furthermore, it contains macros et cetera common to `(l)tugboat.sty`. The list of contents reads

– helpful shortcuts
– abbreviations and logos
– utility registers and definitions
– section heads
– registration marks
– miscellaneous useful stuff
– dates and other items which identify an issue
– issue number in file name
– authors, addresses, signatures
– hyphenation examples
– contents
– change history.

Some 900 lines.

`tugboat.sty` contains

– fonts
– page dimensions
– headers/footers
– page adjustments
– output
– general mechanisms for tags
– titles, authors, addresses
– heads
– text and subtext
– lists
– verbatim
– figures
– utilities
– initialization
– history of changes.

Some 2300 lines.

`ltugboat.sty` contains

– prevent double loading
– fonts (accounts for plain LaTeX and NFSS)
– normal tugboat dimensions
– lists
– titles, authors, addresses, signatures
– Customing of headings to look more like tugboat
– footnotes
– figures
– quotes
– bibliography
– running heads
– OTR
– selfdocumenting style
– miscellaneous defs for compatibility with `tugboat.sty`
– change history.

Some 600 lines.

[21] A bit of wishful thinking—or planning ahead, or whatever you like to call it—for the moment because the functionalities provided differ much. In `ltugboat.sty` it is stated for example 'Redefine style of section headings to look more like *TUGboat*. Start with definitions from `art10.sty`. (Only `\section` so far.)' See also the table 'Functionalities: `(l)tugboat.sty`'.

[22] Sigh, not enough time and/or volunteers. The wish list is unknown to me. If known it would help possibly, apart from the difficulties to adhere to the TUG coding discipline.

[23] This list of abbreviations and logos—with the coded formatting conventions—can better be provided as a separate file to be used by the TeX-community at large, also with formats different from *TUGboat*.

As can be seen the latter two files are not similar. From a logical point of view more can be split into the `cmn` file, especially fonts coupling and headers and footers handling.

I consider the selfdocumenting style as a hobby horse, out of context.

1.6.2 One-ness

Now and then the wish pops up to include all the style files in one system, to encapsulate codes and documentation, better known as literate programming nowadays. Włodek Bzyl and Thomasz Przechlewski just did so by transforming it into `WEB`. (See their TUG '93 article.) The bonus which comes with this approach is that one can easily include change files—they will also become part of the set, and the maintenance annoyance is reduced. An index is by default available, which makes the looking up of commands easier, in principle. At the moment it is unclear to me, within the `WEB` approach, how for example the 'abbreviations' module can be made separately available, to be used in other contexts without the overhead of the total system. It is even darker for me how to make the verbatim functionality separately available. With respect to the customing I agree that the change file concept is beneficial and worthwhile, and in general I believe in the approach. Barabara Beeton communicated that work is in progress to provide it all with `doc-option`, Mittelbach's style option to format documentation of style files. Puzzling is still why not as a `WEB`, as suggested by the polish GUST at Aston last year.

1.6.3 For editors only

It would be nice to organize the style files such that there is a part marked 'for editors only.' And why not indicate in the same spirit parts 'to be removed/replaced in version such and such,' and the like?

1.7 Conclusion

A first-order conclusion is that `tugboat.sty` and `ltugboat.sty` provide superficially the same appearance in print, but differ in the markup and the macros used: `tugboat.sty` respectively LaTeX.

For the communications of a user group the refereeing of *TUGboat* submissions is misplaced.

Provide authors not only with the good users' guide but also with a template with the obliged items in the right order. And what about a simple preprint style?

Provide a list of abbreviations (logos) as a separate file to be used with any format or style.

With respect to developments my wishlist for `tugboat.sty` is

- provide specifications, as a blue print for development
- reshape the style files into a `CWEB` with documentation and index encapsulated
- provide a preprint style (for a proposal see Appendix C)
- allow for customings via change files
- general two↔one-column switching
- generic *simple* bibliography tools (like my BLUe's Bibliography), better still maintain the pre-formatted database at the editor's site, with authors only providing entries to this database
- to make *simple* verbatim functionalities separately available for use with AnyTeX, especially LaTeX
- provide automatic numbering schemes
- symbolic cross-referencing[24]
- common fonts coupling
- keywords and abstract environments
- provide emphasize tools for in-line, for example `+...+` with `\eminline`...`\endemphasize`, and in display, for example via the use of `\preemphasis` and `\postemphasis`
- provide a picture environment, for example start from the macros as supplied in `gkpmac`
- document the use of encapsulated PostScript.

With respect to `ltugboat.sty` my wish is to have that more compatible with `tugboat.sty`, that is allowing a LaTeX user *interface.*

1.8 What does the TUG Annual Meeting proceedings look like?

Great, especially the last one—1993—set in Lucida. A main difference with the regular *TUGboat* is that the titles with *complete* author(s) information span two columns (left justified), along with the (centered) abstract. (Keywords are tacitly discouraged as I experienced. Mine were commented out?!?) The appendices are in one-column and start on a new page. Page numbers and conference information are set at the bottom of the page. (`\midpage` has been recoded in order to allow for running feet.) Footnotes are set per column as in `tugboat.sty`. The headers and footers are different—contain other information—from the regular *TUGboat* issues.

1.9 Processing proceedings TUG Annual Meetings

Basically the same procedures are followed, but handled each time by a different team of proceedings editors. As can be expected there is not much room for experience to be built up, with all respect to the editors who have done a tremendous job in the past.

Refereeing is done too. For the TUG '93 Annual meeting I was surprised to experience that the papers were first refereed and after being accepted there was a second refereeing process—with different referees—in order to warrant the quality for the submission for the proceedings?!? Perhaps the confusion arose because I had the papers already finished when only abstracts were requested. I should have known better, not to submit at that time the paper

[24] I know of Spivak's—LAMS-TeX—rich, independent and general referencing scheme. Włodek Bzyl communicated that the macros used for formatting 'Concrete Mathematics'—gkpmac—also embody such a macro tool, which can be used along with AnyTeX, and therefore `tugboat.sty`. There, however, | is used and we can't use them as such together with `tugboat.sty`.

too. However, there is much miscommunication here. For EuroTEX-s I had to judge 'contributions' submitted as abstracts, complete papers, and even non-submissions?!? The point I like to make is that there is much confusion around and when an author expresses his misunderstandings accept that as *a reality*, whatever the intentions or setup. Accept and say, OK how can we cope with this, it has not been intended, instead of rejecting and saying it is not true.

The proceedings styles are less stringent coded (with all respect). Especially, those which build upon `tugboat.sty`. The latter don't follow the conventions—in full generality—of the tags. Apparently these design conventions have been considered as overdone. The needed functionality has been provided for in a simpler way, superficially similar. See the '`tugproc.sty`' part.

`tugboat.sty`

2 TUGBOAT.STY

Given the users' guide the use of the commands (see Beeton and Whitney, p.379 for the outer form commands, and p.383 for a command list summary) won't give much rise to difficulties in using them. Good work! The version discussed is 1.14, Feb 93.[25]

The `tugboat.sty` is strong with respect to handling of (semi-transparent) verbatims especially the file verbatim (reading and writing), the possibility to use short command forms, the handling of options, and the possibility to delimit the argument by `*`-s or just the end delimiter `\end<foo>`. The optional command handling does not check. It functions like a mechanism for including user guidance.[26] This style should be strong—and undoubtedly it will be in near future—with respect to switching from two-column into one-column format and vice versa.[27]

2.1 Customing

2.1.1 Page size

The page size parameters involved are summarized below (restricted to one and two column)

```
\normalcollgt=54pc \collgt\normalcollgt
\pagewd=39pc
\twocolwd=18.75pc \intercolwd=1.5pc
```

2.1.2 Headers and footers

Headers and footers are set by invoking `\rtitle` and `\rfoot`, respectively, with as replacement text `\hbox to\pagewd{...}`.

2.2 Coding, or TEXies at work

I can't agree more with[28]

> '. . . The tags whose use we encourage are the higher level tags that mark the logical document structure. . . . '

However, there are a lot of questions: whether the right a priori choices have been made, whether the goal has been attained and whether the coding used is the best? Is it easy to adapt, to extend? Can we learn from it? Should it be followed? In order to answer these questions let us consider some elements in detail.[29]

2.2.1 Fonts

Plain's handling of fonts has been adopted. From the file (abridged)

> . . . The fonts—5-10pt, roman, slanted, bold, teletype, extension (math)—are sufficient for most ordinary *TUGboat* productions. Additional titling fonts are defined elsewhere, and occasionally an extra font will be needed for a particular item (e.g. device charts) and defined in that file. . . .

With respect to the recent NFSS the following

> '. . . Anticipated changes to this font handling scheme: Dynamic loading of fonts, probably in groups according to size. . . . '

From the TEX coding point of view the adding to a toks variable is interesting. It is based on the toks extension—`\t=\ea{\the\t *}`—as given on p.373–374 of the TEXbook.

```
\def\addto#1#2{\csname @addsto\string
   #1\endcsname=
     \ea{\the\csname @addsto\string
        #1\endcsname#2}}
%with
\def\@additionsto#1{\ea\the\csname
   @addsto\string#1\endcsname}
```

With, for example, extension via

```
\addto\tenpoint{\def\ssf{\tenss}}
%and use via
\@additionsto\tenpoint
```

Quite something to have a backslash as part of the name of a toks variable!

[25] Still current of Februari 1994.

[26] Ron Whitney communicated that the idea behind this not checking of the user supplied option against the allowable options, was to allow freedom for the user, to include whatever he/she wishes only restricted by that it shall be done via the option gate.

[27] The versions I studied did not contain it in full glory, yet. In MHO a two-column format must allow for tables or pictures which are page-wide, entailing switching to one-column and back. As said elsewhere I like Knuth's approach: one-column the format to work with and now and then provide for pseudo two-column format as implemented in his manmac macros, as demonstrated by the TEXbook itself. This is simpler and more efficient. A matter of taste? (GUTenberg cahiers switched from the two-column style into one-column style last year or so. NTG's MAPS is two-column biased. The EuroTEX '88, '91 and '92 proceedings adopted one-column.) Definitely context dependent. Some people have strong opinions about the optimal width of lines of text. The format for scientific books is different from the format used for journals (and newspapers). The latter prefer multi-column format.

[28] It is a 'white lie,' only part of the truth. We also need 'markup in the small' for example for math, tables and graphics.

[29] Don't look for the answers—if any they are implicit—meaning you have to judge yourself.

2.2.2 Conventions

`tugboat.sty`'s systematics manifests itself in the general mechanisms for

- the markup via begin tags with the short delimiters—`*`-s—or the end delimiter `\end<foo>`
- keeping track of the environments via `\CurrentTag`[30]
- the systematic invocation of the execute macro—logically `\@begin<foo>`—via

```
\csname@begin\CurrentTag\endcsname
```

- the provision of defaults and how to override them, for example

```
\@itemtag={$\bullet$}
\def\tag#1{\@itemtag={#1}}
%with use
\tag{<your itemtag>}
```

- eliminating superfluous spaces and blank lines from input, for example via the use of[31]

```
\def\DeleteOptionalSpacesandPars#1{%
  \@ignoreall{ }{\@ignoreall{\par}{#1}}
```

- handling of optional arguments (attributes)
- the inclusion of `\every...`, for example, the powerful `\everyverbatim{\enablemetacode}`, to handle metacode in verbatims throughout
- general attributes (for example `\ruled`, `\numbered`, to be orthogonally used with various constructs)
- the handling of the separation with what follows after an element via `\@next`
- the two ways of handling an 'argument': via storing in `\@argument` and use it later, or on the fly (the latter—characterized by `\@savingargumentfalse`—has the advantage that the catcodes have not been set).

Great!

Outer form tags are implemented in a *consistent* and orthogonal way. All the tags obey the same syntax. After the begin tag optional parameters can be used independent of whether the short-form delimiters or the end delimiter is used. Quite a TeX coding marvel! However, the negative side is the too rigid discipline.[32] Even for simple things like `\head*...*`—where at the user level there is no much need for optional parameters—the rigid discipline has been adhered too.

> In this case, however, all what has essentially to be done is typesetting of the head title in an appropriate typeface with sufficient white space around it, and paying attention to the right penalties in order not to let the head title start at the bottom of a page.

So, this beautiful discipline has been applied at the expense of simplicity and flexibility now and then. Which is quite understandable. Furthermore, I for one don't consider the coding of the title part relevant. Of course, the creators of the styles have to worry about it, but I won't. Everybody rides his hobby here. I use templates for these parts given by the various sources, whatever the codings.

2.2.3 General mechanism for tags

An author can choose from the following variants to mark up his copy with respect to the tag `\<foo>` via

```
\<foo>*<argument>*
\<foo>[<options>]*<argument>*
\<foo><argument>\end<foo>
\<foo>[<options>]<argument>\end<foo>
```

with spaces absorbed where necessary, especially around `*`-s. From the file `tugboat.sty` the following background to the process flow has been copied.

Upon sensing an opening tag (call it `\<foo>` here), the following process is set in motion.

1. `\begingroup` (so definitions and settings are localized).
2. The default situation for `\<foo>` is set up.
3. If appropriate `\every<foo>` list is read (this allows one to override the `tugboat.sty` factory defaults).
4. Optional commands are read, the '\', '{', and '}' are restored to their status appropriate to `\<foo>`.
5. The `\@begin<foo>` macro is executed. This may involve branching dependent upon flags set by options. It may also be a place where spaces and carriage-returns are activated.
6. The 'argument' to `\<foo>` is read and stored or processed on the fly (the method employed is generally fixed for each tag). The argument may be delimited by *...* (called the 'short-form' here), or supplied up to ...`\end<foo>` (called the 'long-form').[33]
7. A cleanup macro is executed which also ends the current group. This may do all the work if an argument has been read and stored.'

From the users' guide the following operational scheme has been copied.

```
<read tag>
\begingroup
<set defaults>
\the\every...
<read options>
<branch to appropriate action,
    using 'argument' as necessary>
<cleanup>
\endgroup
```

From one point of view the uniform coding scheme is attractive: once understood, one can read the TeX coding of all the tags. Hmmm, is that true? For me it is not true, because it is too difficult to really understand the coding. It costs too much time or energy to master, while

[30]This is used for the invocations of the specific `\every`⟨*foo*⟩, `\@begin`⟨*foo*⟩, and `\@end`⟨*foo*⟩ commands. With elaboration via the processing on the fly branch there is no check on the correspondence of the `\begin`⟨*foo*⟩ and `\end`⟨*foo*⟩ tags.

[31]This looks strange especially in view of plain's `\ignorespaces`, and that spaces are deleted after control sequences anyway. Ron Whitney communicated that what comes after is executed, and that is needed now and then. Hmmm?!?

[32]Apparently too difficult for epigons, as demonstrated in customing `tugboat.sty` into `tugproc.sty`.

[33]Quite something to do this without the parameter separator mechanism! Moreover, this handling via parsing instead of via the parameter separator TeXnique is more powerful, because it can gobble optional spaces and pars.

the alternatives are simpler, albeit less uniform. Relevant questions are: Should I really work it through? Understand it thoroughly? Communicate about it? A little pondering prompts me that there is hardly an audience for it except me, despite the smart coding.[34]

> From another point of view the advanced TEXnology hinders the extension of `tugboat.sty` once the developers are out of touch.

I will try to stay at the outer level and discuss the global flow of what is going on and distill as many TEX programming paradigms as possible.

2.2.4 Checking ahead

This mechanism is amply used in the `tugboat.sty`. The comments in the style file read

'. . . Often we check ahead to determine the next course of action. `\@checknexttoken` is used to check for optional commands, to check for the short-form argument-delimiter, and to ignore characters in certain situations. The macro is just a check; applications must do whatever is appropriate with the ensuing tokens.

`\@checknexttoken`—with three arguments—checks the next token against argument #1. If the two are the same, #2 is executed, otherwise #3. The comparison is done with `\ifx`. Since we check ahead with `\futurelet`, the first argument is stored with `\let` as well. To include the case where #1 may be a space we have to go through a small contortion to `\let\@basetoken`= that space. . . . '

`\@checknexttoken` is invoked *directly* in

- `\def^^M`
- `\@afterbegintag`
- `\@checkoptions`
- `\@ignoreall`, `\@ignoreone`
- `\@firstverbchar`
- `\page`.

These kinds of macros are important in TEX coding because the course of action is determined by the tokens in the input stream, read by the markup. Spaces are unintended now and then so we like to get rid of them. In order to understand the macro the following from the syntax

```
\let<control sequence><equals>
    <one optional space><token>
\futurelet<control sequence><token><token>
%and from tugboat.sty
\long\def\@checknexttoken #1#2#3{%
  \futurelet\@basetoken\iffalse#1\fi
  \long\def\@next{%
    \ifx@baseis@next
      \long\def\@@next{#2}%
    \else\long\def\@@next{#3}\fi
    \@@next}%
  \futurelet\@nexttoken\@next}
```

Illustration of function `\@checknexttoken`

Toks after	Match?
abc a	no
xyzx	yes
{ }bc b	yes
{ }bc{ }bc	no!

The results in lines 3&4 in the table look strange but come from the nature of scanning by tokens. The simplified (educational version of the) macro reads

```
\def\chneto#1#2#3{\let\@bt= #1
   \def\@next{\ifx\@bt\@nexttok
           \def\@@next{#2}%
      \else\def\@@next{#3}%
      \fi\@@next}%
   \futurelet\@nexttok\@next}
```

Explanation. The macro adheres to the usual structure for looking ahead: a `\futurelet\@nexttok\@next` at the end with the process macro `\@next` defined in the replacement text. The first difficulty is the way the first argument is compared to the `\@nexttok`. I simplified the assignment into the use of `\let` instead of `\futurelet`, with the unnatural kludge `\iffalse#1\fi` removed. Second, the documentation says there is a problem in scanning when an outer control sequence is hit. The production macro solves this by introducing an extra level via `\def\ifx@bt@nxt{\ifx\@bt\@nxt}`. A TEXing paradigm!

2.2.5 DeleteOptionalSpacesandPars

From the file I borrowed

```
%Execute #1 after ignoring spaces
\def\DeleteOptionalSpaces#1{%
   \@ignoreall{ }{#1}}
%Execute #1 after ignoring spaces and \pars
\def\DeleteOptionalSpacesandPars#1{%
   \@ignoreall{ }{\ignoreall{\par}{#1}}}
```

On first sight it seems that there are superfluous curly braces. However, the definition takes two arguments. Apparently there is a nested invocation. Pondering about the macros reveils that there is an 'error-correcting' process inserted too. The parameter #1 is known at design time. So, whatever follows, this #1 is executed in reality. Hmmm, is this a really beneficial side-effect? Barbara Beeton communicated that '. . . this macro permits for a blank line to be left after every section heading of whatever level without having to worry whether it is 'paragraphed' or run in. This is provided almost entirely for the sake of a simple and robust user interface, one that is forgiving of reasonable variations in user practices. . . . '

The following is a comment from Ron Whitney which shows the subtleness involved.

> '. . . The `\@ignoreall{ }` functions somewhat differently than `\ignorespaces` in that it gobbles all spaces and then executes a given command (as in `\@ignoreall{ }{\<foo>}`). If `\<foo>` itself takes arguments, one wouldn't want

[34] When I finished working on this paper, I realized that there is definitely a need for these kinds of discussions and clarifications. New TUG annual meeting proceedings editors should peruse the paper and also study the `tugboat.sty`. An editor should understand that style as a prerequisite, how about that? (No new editors anymore?)

"\<foo>\ignorespaces" in place of this. Perhaps you're—me thus—thinking that \<foo> will ignore spaces anyway in searching for an argument, but this depends upon how \<foo> is defined, and I felt the other approach was uniform and provided something stable from which to work.. . . '

Pondering about the desired functionality I can imagine that the problem arose in implementing the scanning of a stream of tokens with in certain contexts a loose syntax with respect to spaces and blank lines, for example around *-s. I have nevertheless the feeling that \ignorespaces would do—spaces between \<foo> and argument?!?—especially when augmented with a more rigid syntax in order to get and maintain as simple code as possible. This despite respecting 'worn-in' user habits, for which we should always have an open eye and ear. And, after all an author should always *mind the spaces* with TEX markup.

Whatever your opinion and for the fun of it I have a few alternative codings

```
%1: skip until #1 and execute #1
\long\def\deleteuptoandexecute#1{\long
   \def\gobble##1#1{#1}\gobble}
%2: just gobble #1 repeatedly
\long\def\gobbleall#1{\long\def\fifo##1{%
    \ifx#1##1\else\long\gdef\nxt{##1}\ofif
    \fi\fifo}%
   \fifo}
\def\ofif#1\fifo{\fi\nxt}
```

Remark. The above is straight and works for copy *without outer defs*. The latter can't be used with fifo with an argument. As Spivak found out earlier outer defs give a lot of 'pain,' read complicated codings. Below—for hackers only—I have enclosed a teaser which allows outer defs to be scanned. It has been tested to delete spaces and pars. My first significant application of \afterassignment.[35]

```
\let\aa\afterassignment%local shortcut
\long\def\gobbleall#1{\let\nxt\relax
  \let\arg= #1
  \def\fifo{\ifxan\else\ofif\fi
     \aa\fifo\let\nxt}
  \aa\fifo\let\nxt}
\def\ifxan{\ifx\arg\nxt}
\def\ofif#1\let{\fi}
```

A parameterless version is

```
\def\gobbleall{\let\nxt\relax
   \def\fifo{\def\fifo{\ifxan\else\ofif\fi
                       \aa\fifo\let\nxt}%
             \aa\fifo\let\nxt}%
   \aa\fifo\let\arg}
\def\ofif#1\let{\fi}
%with test
marknew\gobbleall\newtoks\newtoks
\newtoks after new
```

A nice addition to my list of applications of the FIFO paradigm.

2.2.6 Options

The options are parsed via |checkoptions|,[36] which at some stage invokes \@@readoptions[#1]{#1...}.

Note that the replacement starts with #1, so 'the option' is just inserted,[37] and there is no checking!

Aha, if so there is a much simpler TEXnique for this optional parameter functionality. Namely, in analogy with

\every<foo> introduce \this<foo>!

This approach is worked out along with the alternative for handling verbatims (see later on).

The \@checkoptions copes with repeated options via recursion by invoking \@checkoptions again. This process is stopped when no [is sensed.[38] It is complicated because the code has to look ahead for options and after that for the short-form separators or the end delimiter. And last but not least it is also complicated because the options must be processed with the right catcodes. When BLU dares to delve into the code he might perceive from \setupverbatim that the backslash has gotten category code 12 (other) and that therefore the commands supplied as optional arguments won't work. (In the code \@SpecialsGetOther precedes \@checkoptions!) However, tugboat.sty's comments say

'. . . Since initial setup involves changing the special characters to characters of type other, some juggling must be done when optional commands are read. . . . '

From the file we have '. . . \catcodes of \ { } are restored to their plain values . . . ' via

```
\restorecat\\
\restorecat\{
\restorecat\}
```

Too difficult for BLU, for sure.

And finally via \@executetoend the
\csname @begin\CurrentTag\endcsname
that is the \@begin<foo> macro is executed.[39]

A poly-algorithm, and certainly not a UNIX filter to be combined easily with other filters. But uniform and consistent it is, for sure, if not for being so un-unusual.

2.2.7 Sectioning 'commands'

Actually these are the markup commands for the headings. \head, \subhead, and \subsubhead all have

[35] For a production version don't use the shortcut.

[36] As follows from below this command does much and much more than just inserting the options. It also executes to the end!
And what about '\futurelet \@basetoken \iffalse #1 \fi'? Perhaps naive, but I think the above is equivalent to \let\@bastoken= #1. However, it is ingenious in that it removes #1 from the input stream after the use of \futurelet.
Salomon in his courseware uses basically the same approach with simpler coding, though. He just inserts the options and performs the catcode change of the backslash after that.

[37] Mind the catcodes! Agreed this is pin-pointing to details, but as my teacher T.J Dekker would say: details matter.

[38] For the exceptional case that the verbatim text starts with a [, then \lastoption can be used which sets \@lastoptiontrue.

[39] A consequence is that it is not straightforward to search the file for the use of a specific \@begin⟨*foo*⟩.

the same ending `\endhead`. (I have never used `\endhead` in practice, always the `\head*...*` form.[40])

'Heads are set by first saving the text of the head in `\@argument`[41] and then operating appropriately depending upon the `\headlevel`. Selection among the different heads is made by an `\ifcase`.'

The typesetting is done by `\@domainhead`, `\@dosubhead`, and `\@dosubsubhead`, for respectively `\head`, `\subhead`, and `\subsubhead`.

> They all come basically down to typesetting of `\bf\the\@argument`.

Only the second (level) adds an end dot if `\if@headpunctuation` is true! (This default dot can be omitted via `\nopunctuation`.)

`\@next` is defined for each case and takes care of how the next line is separated: all (optional) spaces and `\par`-s are eaten, and only occasionally the main head does not allow for indentation. This `\@next` is general and also used after lists and verbatims. A neat convention!

I really can't tell whether a head will pop up in print near the bottom of a page. For a (sub)subhead no penalties have been inserted so they might end at the bottom of a page. Not nice when fully-automated typesetting is strived after.

From the macro file I also stumbled upon
`\sectitle...\endsectitle`
with the underlying `\@sectitle` in `tugboat.cmn`. These are used by the editor to mark up '...the boxed title for the major column headings in *TUGboat*.'

2.2.8 Lists

Lists are preferably marked up by `\list ...\endlist`.[42] After the opening tag optional material might be specified, as usual. The options to choose from are what we call now the attributes: `\numbered`, `\romannumeraled`, `\Romannumeraled`, `\lettered`, `\Lettered`, next to `\ruled` and the specification of the tag `\tag{...}`.

Peculiar—but useful and I like these kinds of parameterizations—is the option
`\def\itemseparator#1{\def\@itemseparator{#1}}`
This will yield a comma as separator (default). However, this is only so when the list is *not* in the default `\displaystyle`. Confusing all those style variants, which by the way are not documented in the users' guide.

TEXnically there is also the attribute `\itemized` (default) and `\unitemized`, where in the latter case the carriage-return separates the items. From the example in the users' guide I get it that when the option `\numbered` is used one also has to be explicit about `\unitemized`. Hmmm, looks error-prone to me. Reading the code reveals that my perception is wrong. With `\unitemized` the item label is updated and inserted via `\everypar`, where every end of line is also treated as an end of paragraph!

```
\everypar={\advance\itemnumber\@ne
          \tagform{\the\@itemtag}}
\makeCtrlMendgraf
```

Apparently too advanced or too much detail in order to be mentioned in the documentation.

The surrounding of the list by horizontal lines before and after is governed by the `\ruled` option.[43] For the moment there are no sublists supported.[44]

The `\everylist` command is handy when one sticks to one kind of lists. For example, the default is a bulleted list and this can be changed into a numbered list via `\everylist{\numbered}`.

One can mimic LATEX's list. For example the itemize environment can be expressed in `\list...\endlist`, as follows.

```
\def\itemize{itemize}
\def\end#1{\ea
   \ifx\csname#1\endcsname\CurrentTag
    \endlist
   \else\message{No matching \CurrentTag
              end}
   \fi\egroup}
\def\begin#1{\bgroup\def\CurrentTag{#1}
  \ifx\CurrentTag\itemize\ea\list\fi
}%with LaTeX's  use
\begin{itemize}
\item ...
\end{itemize}
```

To mimic the other LATEX list environments some more work needs to be done, especially the appropriate branching for, and the correct handling of, the specific situations.

> The above has been included to illustrate the idea of a LATEX user interface on top of `tugboat.sty`.

In order not to lose the wood for the trees I have neglected the enumerate and description environments.[45]

The functionality provided by the optional parameters of the `\list` command is really handy for authors who change their copy often.

They will benefit most from the automatic numbering, be it by numbers or letters.

[40] Note that because the form is parsed the syntax is flexible with respect to spaces around the *-s. This in contrast with the rigidness of the parameter separator TEXnique.

[41] Because of this verbatims will go wrong in the titles of the headings. LATEX's notorious fragile.

[42] This holds especially when the `<Contents>` has te be marked up by it and not so much the `<Title>`.

[43] This is something not special for lists. It can also be used with verbatims (or figures). The same with `\numbered`, to number lines in verbatim texts. However, I consider it better to provide this functionality separately and independently, as one of the ways we like to encapsulate document elements. Why not framed, to name but one alternative in use with tables?

[44] Barbara Beeton commented that these are ranked high on her list of wishes.

[45] Perhaps another time I will come back to it in full generality.

The coding of the options is a bit complicated because of the way alphabetic 'numbering' has to be coded in TEX. The way to do this within TEX is to bias the counter variable—`\itemnumber`—by `"60`, respectively `"40`, and advancing the counter as ususal. With `\@itemtag={\char\itemnumber.}` the right letter will be obtained. A TEXing paradigm! Roman numerals go much along the same lines: `\@itemtag={\romannumeral\itemnumber}`.

Nested within `\list` is `\item` with optional parameter `\tag{...}`.

Alternative
Without optional parameters many alternative codings can be obtained via the use of Knuth's `\item` enhanced with some code from `tugboat.sty`. The `\list` functionality—after abstracting from details—can be obtained via the following progressive coding. For the default, that is the bulleted list, the following.

```
\let\dekitem\item
\newtoks\itmtag \itmtag={$\bullet$}
\def\item{\dekitem{\the\itmtag}}
\def\list{\prefoo\bgroup}
\def\endlist{\egroup\postfoo}
\def\prefoo{}\def\postfoo{}%e.g.\hrule
%with use
\list
\item <item copy>
\endlist
```

The next step is to allow for 'options.' Below I have coded '`\numbered`.' This can be added to the above. Similar things hold for the other options.

```
\newcount\itmnum
\def\numberedlist{\prefoo\bgroup
   \def\list{\itmnum0
      \itmtag={\global\advance\itmnum@ne
           \number\itmnum.}}
   \list}
%with use
\numberedlist
\item <item copy>
\endlist
```

Just the little extra to Knuth's `\item`, to make our TEXing life a little easier. Not too clever, nor too general. Simple, straight and direct, respecting the way Knuth has shown us. The point I like to make is that more could have been built directly upon what Knuth had already provided.

Moreover, the checking of the correspondence of the `\begin<foo>` with the `\end<foo>` is no longer much needed since the advent of the (LA)TEX intelligent editors which prompt you with environments.

It is admitted, however, that `\everylist` does not combine easily with the above.

Intermezzo: prefix versus postfix.
Optional parameters are active between the opening and closing tag, with the option(s) supplied right *after* the opening tag. Loosely speaking this is reminiscent of 'prefix operators.' On the other hand my approach via the opening tag placed *after* the 'option(s)'—as shown in the above alternative—is reminiscent of 'postfix operators.' However, my mechanism yields after expansion a similar execution order. For the example above after expansion the execution processes goes along the lines

```
\prefoo%Whatever you wish to be done
\bgroup
\list%modified with numbered items
\item ...
\endlist%equals \egroup\postfoo
```

End intermezzo.

What confuses me are the more-column lists. What are we talking about? A table? Or do we just like to be economical with the space and cut a long list into pieces set next to each other? Is this descriptive markup? I never used this facility, and think I would not if only for tables. Is there a practical need for this?[46]

2.2.9 Verbatims

Verbatims are provided functionally for

- in-line use, via `|...|`, or `\verbinline...\endverbatim`
- in display use, via `||...||`, or `\verbatim...\endverbatim`

with the following options

```
\inputfromfile{...}
\outputtofile{...}
\numbered
\continuenumbers
\ruled
\lastoption
\makeescape\...
\enablemetacode
```

The best verbatim functionalities (and codings) I have ever laid eyes on. Neat! Note that here a minimal form of markup can be used—by `|`, or `||` for begin and end tag for in-line and display, respectively—this while the `<Contents>` is enclosed by the begin tag and end tag. Next to that the general mechanism of supplying options can be used.

How are verbatims processed? The verbatims are processed via storing *all* the verbatim text until `\endverbatim`—or its alias—in `\@argument`. This is not easy to read from the code because first the opening tag—or its alias—and optional arguments are parsed via *TUGboat*'s mechanism. After that `\@executetoend` invokes `\@beginverbatim` with as replacement text `\obeyspaces` (`\obeylines` is on already.) Finally, in `\@longparse`—or `\@shortparse`—the verbatim text is stored in `\@argument` *with the right catcodes*. This code is advanced and a teaser for those who think they understand TEX. For your convenience I have included below a piece of code—one of the parse variants—that takes care of the parameterization over the end separator.

```
\def\@longparse{\if@savingargument
  \edef\@form{\def\nx\@@longparse####1\the
  \enddelim}%
  \@long\@form{\@argument{##1}%
  \csname end\CurrentTag\endcsname}%
\else\def\@@longparse{}\fi
\@@longparse}
```

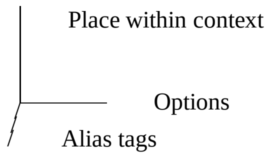
[46] I get it that this feature is taken over from Berry's eplain?

Explanation. In lines 2&3 the concrete end delimiter is inserted via the expansion of the `\edef`. In lines 4&5 `\@@longparse` is defined with as replacement text the storing in `\@argument` of the verbatim text with the right catcodes (remember that `\setupverbatim` is on already) and the invocation of logically `\endverbatim` to typeset the (stored) verbatim text. Quite something isn't it?

Alternative: parameterization over end delimiter. This can be done differently. I used earlier[47] for parameterization over a separator the following two-level approach

```
\def\enddelim#1{\def\readtoenddelim
  ##1#1{\def\store{##1}}}
%with use
\endelim\enddelimiter
\readtoenddelim
%\store contains now all which follows
%after \readtoenddelim up to \enddelimiter
```

Intermezzo: functionalities. In general these kinds of functionalities as provided by `\verbatim` can be grouped into the following orthogonal sets.[48]



By the way the above was hacked via

```
\setbox0=\vbox{\hbox{/}\kern0pt}
\def\xyz#1#2#3{\leavevmode \lower
   2.65\ht0\hbox{\copy0\rlap{\qquad#1}}%
   \kern-.185\ht0\lower1.8\ht0\copy0%
   \kern-.185\ht0\lower0.95\ht0\copy0%
   \kern-.11\ht0%
   \vrule height9ex depth0pt\relax
   \raise7ex\rlap{\qquad#3}%
   \vrule height.1pt depth.1pt width 8ex
   \qquad\lower.6ex\hbox{#2}}
\quote
\xyz{Alias tags}{Options}{Place within context}
\endquote
```

Clarification

- place within context, denotes how the element is set within the context (in-line, display)
- options, denote the parameterization of the functionality proper (here: copy, file, . . .)
- alias tags denote alternative tags (minimal markup, for example by `|`, or `||`).

The coding as provided is *not modular.* The functionalities are mixed up in the coding, also called monolithic or poly-algorithmic code. History has it that these kinds of codes are difficult to understand, to maintain, to extend, or to be ported (partially). If I had to code these sets of functionalities I would do that in an orthogonal way, each set independent from the others.

For example the 'typesetting within context' is not at all specific for verbatims, it applies equally well to other document parts like figures, math, tables, or you name it. Options—or let us call them parameters—are for example to have a display with rules before and after, or to have a frame around the document element as can be seen with tables. So these can better be provided separately in such a way that they can be used orthogonally with other document elements.

With respect to options I would—and have already in the past—adopt Knuth's parameters. Here I played with the idea to provide `\thisverbatim`, in analogy with `\everyverbatim`, as is touched upon a little further.

I like to look at alias tags as extras which are virtually absent when not used. With verbatims there is a problem because of the fixed category codes. As shown in `tugboat.sty` one can parameterize over the end delimiter.

With respect to coding separate verbatim user macros for each functionality I consider that not economical because we like orthogonal combinations of the functionalities too. To provide codings for all these combinations would require too many macros.
End intermezzo.

Knuth's verbatims. If we compare the above with the functionalities already provided in the TEXbook, Appendix D: 3 *Verbatim listing* then the additional mechanisms are

- the optional parameter mechanism provides choices to be made by the user (In The TEXbook Appendix D the file verbatim is numbered by default.)
- the writing to a file verbatim[49]
- the definition of an escape character for semi-transparent verbatims.[50]

However, in typesetting the CWEB manual Knuth&Levy introduced the following *verbatim mode*[51]

```
\def\verbatim{\begingroup\dospecials
   \parskip0pt \parindent0pt \let\!=!
   \catcode‘\ =13 \catcode‘\^^M=13
   \tt\catcode‘\!=0 \verbatimdefs
   \verbatimgobble}
%with auxiliaries
{\catcode‘\^^M=13 catcode‘\ =13
 \gdef\verbatimdefs{\def^^M{\ \par}%
    \let =\ }
 \gdef\verbatimgobble#1^^M{}}
%with for example file verbatim via
\def\tt{\eighttt}\baselineskip9pt
\def\printmacs{\input cwebmac}
\verbatim
!printmacs
!endgroup%ends verbatim
```

[47]Locating a character in a string, or a card in a (bridge) hand, see my Syntactic Sugar paper.

[48]For the poor man's slanted axis I first tried to box the / from the calligraphic font and was quite surprised to find a box of width 0!

[49]This has been coded in the TEXbook in the `\answer` macro. There the text will be written to `answer.tex`.

[50]This mechanism is handy with verbatims which contain pieces to be set in a font different from `\tt`, as worked out in the macro `\enablemetacode`. When this is used as option with verbatim the metacode can just be specified by `<metacode>` with result $\langle metacode\rangle$. Handy!

[51]With ! to be marked up via !!.

Agreed Knuth&Levy are by no means average TEXies, but it gives food for thinking when a verbatim package is complex.[52] In essence the structure is the same as with my `\thisverbatim`, differing modulo some syntactic sugar. Perhaps my approach is a nice compromise between simplicity and abstract rigorous markup, with as extra bonus no parsing overhead.

Another nice insight, due to Rainer Schöpf, is that verbatims and block comments are closely related. Block comments don't format, they just skip copy!

Intermezzo: block comment.
In practice I have used permanent and temporary block comments. The first is used for documentation (of code/macros), and are marked up in TEX simply by starting each line with the %-character. The temporary comments are needed now and then to locate markup unbalances by commenting out document parts. A way how to do this is to let your editor insert %-s. Enclosing the document part(s) by `\iffalse` and `\fi` does not work when outer defs are enclosed (`\beginsection`, `\new<foo>`, . . .). From a user point of view it is nice to provide `\comment` and `\endcomment`. Later on block comments will be treated as a special case of verbatims, so I'll refrain from targeted code.
End intermezzo.

A TEXing paradigm is the parameterization over the end tag symbol the 'end verb delimiter.' First, we need '. . . to 'see' the tag when '\' is of type 'other'. . . ' via

```
%In \setupverbatim
\enddelim=\ea{\endverbdelimiter}
%with the default
{\catcode‘\|=0 \catcode‘\\=\other
 |gdef|endverbdelimiter{\endverbatim}}
```

Second, we have the specifics for the `|` (and similar for the `||`). This has been accounted for via

```
\makevertverbchar%at end tugboat.sty
%which defines
\@verbchar={|}
%and invokes
\setupverbchar
%the latter defines among others
\catcode\ea‘\csname\ea\string
  \the\@verbchar\endcsname=\active
\edef\endverbdelimiter{\the\@verbchar}
\enddelim=\ea{\endverbdelimiter}
```

Verbatims seem to need 'processing on the fly' code. At the outer level this is so—no catcodes fixed—but at a lower level after the catcodes have been appropriately defined the verbatim `<contents>` is stored. This explains `\@savingargumenttrue`.

> The big disadvantage of this is that long verbatims on small TEX systems will yield the error message 'Capacity exceeded!'

In order to understand the various processes the reader is advised to begin with the TEXbook p.380, where for example the details which come with inclusion of a file verbatim (with line numbering) has been discussed starting from

```
\def\listing#1{\par\begingroup
  \setupverbatim\input#1 \endgroup}
```

The cornerstone is of course `\setupverbatim`. For the coding of writing to a file verbatim the reader is referred to the `\answer` macro, which comes with `\exercise`, TEXbook, p.422. Another educational inroad to the world of (file) verbatim intricacies is provided in Salomon's courseware.[53]

In the *TUGboat* authors' guide the writing to a file has been illustrated for 'storing' address information—marked up at the beginning—and used at the end. In MHO—with all respect—this is a bit misplaced example, because this information could have been stored in token variables or so. More relevant examples deal with when moving information which needs catcode changes later on. Knuth did this with his `\answer` macro, his writing of index reminders, and I can think of the handling of endnotes as being relevant too.

So the conclusion is that Knuth's codings have been adapted to the general *TUGboat* coding schemes—especially obeying the orthogonal and systematic handling of optional parameters—next to the incorporation of some fine-tunings. I missed however the negative `\disablemetacode`—to switch it off temporarily—especially when we use the `\everyverbatim`{...}-s.[54]

Alternative: the big deal. At the heart lies the wish to have verbatim functionalities available for use with AnyTEX. As it is provided now it is too complex to be used within other contexts than `tugboat.sty`.[55]

It is admitted that verbatim facilities are mostly needed by authors who write about (LA)TEX. The approach to classify facilities in orthogonal sets is fundamental though. So hang on. I'm not claiming that I have provided a production version, but I hope my approach will be recognized as being simpler. To illustrate my ideas I will treat below

- place within context, in-line versus display via `\preverbatim` and `\postverbatim`
- options, or parameterization via `\thisverbatim`
- alias tags.

In-line versus display. The whole idea is to concentrate on *one* verbatim functionality with markup structure

```
\verbatim
..
\endverbatim
```

that will handle them in general with no extras, just in-line, with `\obeylines` on, which is superfluous with really short verbatims, as a special case. The markup is always

[52]If only I was aware of this when I submitted my first *TUGboat* article—Typesetting Bridge, when verbatim file inclusion was not yet available—my TEXing life would have been simpler.
[53]Or consult his article on the issue submitted to *TUGboat*.
[54]Just a very minor issue, which can be settled via the escape character functionality.
[55]This is characteristic for monolithic code.

in this form, with \endverbatim on a separate line. Then the only thing we need for displays is to enclose it by something like \medbreak-s! This can be parameterized eventually in \preverbatim and \postverbatim. Earlier I used this approach with respect to loops, see my Syntactic Sugar paper. This was what I had in mind, but in reality it worked out differently: first the display version and its variants, and later as a special case the (alias) in-line version. Not complete nor general. Sufficient and simple. Just what I need. Hang on.

Options via \thisverbatim. Basic is the idea of the toks variable \thisverbatim in analogy with Knuth's \every<foo>. To illustrate my approach I selected the following verbatim functionalities

- a user defined escape character for semi-transparent verbatims
- verbatims with lines numbered (TB, p.380, 381)
- input a file verbatim (TB, p.380).

First of all how would the user interface—the basis for any specs—look like? Is the following simple enough?

Examples of use.

```
%1. To handle metacode and font changes
\thisverbatim={\emc%enable metacode
               \escapechar\!}
\verbatim
Some <meta code> and
blah, blah, ...    !it
Now text in italics!tt
and back again in tt
\endverbatim
%
%2. To handle numbering and verbatim file
%   inclusion
\everyverbatim={\numvrb}
\thisverbatim={\input \jobname.tex
               \input vrb.tex}
\verbatim
Extras to be set (verbatim) after file
\endverbatim
%
%3. To handle block comments
\thisverbatim={\blockcomment}
\verbatim
Block comment
to be deleted/skipped.
\endverbatim
%
%4. To restart (line)numbers
\thisverbatim={\vrblin0 }%or supply \numvrb
\verbatim
Just some text with
line numbers restarted.


Text after two blank lines.
\endverbatim
```

For in-line verbatim I decided not to supply another version than via the alias tags because that is what is needed most of the times and simple. Hang on.

The coding alternative is based on TB p.382, modified with line by line processing via the fifo paradigm, van der Laan, 1993. The multi-level coding approach resembles the way how I have built the sorting macros, see my Sorting in BLUe. The macros are available at the CTAN-s in directory macros/generic/vrb.

```
%Macro codes:
%User toks variables
\newtoks\thisverbatim
\newtoks\everyverbatim
%
%User Customing
\let\preverbatim\medskip
\let\postverbatim\medbreak
%
%User 'options'
%\numvrb
\def\numvrb{\vrblin0
   \everypar{\advance\vrblin1
    \llap{\sevenrm\the\vrblin\quad}}}
%\blockcomment
\def\blockcomment{\def\processl##1{}%
    \def\preverbatim{}\def\postverbatim{}}
%\emc%my simplified enable metacode
{\makeactive\<
 \gdef\emc{\makeactive\<%
   \def<##1>{$\langle##1\rangle$}}}
%\escapechar\...
\def\escapechar#1{\catcode`#1=0 }
%
%User macro (\endverbatim implicit)
\def\verbatim{\preverbatim\begingroup\tt
   \setupverbatim
   \the\everyverbatim
   \the\thisverbatim
   \verbatimgobble}%\fifol for all but first
%
\def\verbinline{\bgroup
   \def\preverbatim{\let\par\relax}
   \let\postverbatim\egroup
   \verbatim}
%%%%%%%%Lower level macros%%%%%%%%%%%%%%
\def\makeactive#1{\catcode`#1=13 }
%
%Default
\def\processl#1{#1\par}
%
\newcount\vrblin
%
\def\setupverbatim{\makeactive\`%TB381
   \def\par{\leavevmode\endgraf}\obeylines
   \uncatcodespecials\obeyspaces}
%
{\makeactive\` \gdef`{\relax\lq}}
%
\def\uncatcodespecials{\def\do##1{%
   \catcode`##1=12 }\dospecials}
%
%For FIFO see TUGBoat 14, 1, EuroTeX '92
%proceedings, or MAPS 92.2
{\catcode`\|=0 \catcode`\\=12 |obeylines%
 |gdef|ev{\endverbatim}%
 |gdef|fifol#1
    {|def|tst{#1}%
     |ifx|tst|ev|lofif|fi%
     |processl{#1}|fifol}%
 |gdef|lofif#1|fifol{|fi|endgroup%
    |thisverbatim{}|postverbatim}%
 |gdef|verbatimgobble#1^^M{|fifol}%
 |obeyspaces|global|let =| }
```

Promising and simple. It provides most of what I need. Perhaps it should be worked out with all of the ornantia as provided by tugboat.sty. For writing to a file verbatim—needed for writing index reminders, or answers to exercises—I would start with what Knuth already provided, . . . and stay close to it.

What I like to get across is that the same functionalities can be obtained with less complicated coding, at least with more flexible code.

Alias tags. The opening tag is not a problem. And in general neither the closing tag. But with verbatims it is a bit difficult.[56] The specs could read

```
%5. In-line verbatim with alias tags
\thisverbatim{\emc}%enable metacode
Before |in-line <text>| after.
%with macros
{\makeactive\|
 \gdef|{\bgroup\tt\setupverbatim
    \the\everyverbatim
    \the\thisverbatim
    \def|{\egroup%From manmac
          \thisverbatim={}}}
```

Hmmm, some work to be done. Given what is already available, it will have low priority. For the moment I stopped and will look over BLU's shoulder how things will go in practice.[57]

2.3 Placement of figures

Placement of figures are accounted for via the markup

```
\figure
<vertical mode material>
\endfigure
```

with as result 'a single-column floating top-insertion.' The options implemented are

```
\top%default
\mid
\bot
\caption{...}
```

Peculiar is the provision of the caption via the token variable `\caption`, as option.[58]

This stretches the option concept. Not only the representation is the subject of an option but also whether the components appear in print at all.[59]

When two-column formatting is the default we need mechanisms to handle 'figures spanning columns.' With respect to the latter the following is borrowed from the users' guide.

> 'Figures spanning columns at the top and bottom of a page are currently supported only on the first page of an article, but we expect they will soon be allowed on any page (a general rewrite of the output routine is in progress). `\twocolfigure` (terminated by `\endfigure`) starts up such a figure and currently *must* occur before any material has been typeset on the first page (i.e. before `\article`).'

With respect to intermixing `\onecol`, `\twocol`, and `\threecol`, the following from the users' guide.

> '... these can't currently be intermixed on a page.'

By the way the table about the functionalities of `(l)tugboat.sty` in the beginning of this paper, was set via

```
\figure[\mid]
$$\fll\btable\data$$
\endfigure
```

with `\data` and `\rowstblst` appropriately filled, and `\caption` of `\btable` adapted into `\capbtbl`, because of name clashes.

2.4 Argument processing on the fly

LaTeX's footnote first stores the footnote text while plain's footnote processes the 'argument' on the fly. In practical terms this means the we can use verbatims in plain's footnotes and not in LaTeX's.[60] I consider the coding of argument processing on the fly a TeXing paradigm. The example of how to do this is provided in the TeXbook by the coding of `\footnote`, p.363. That footnote coding buries the 'processing on the fly' TeXing paradigm, because necessary details had to be accounted for, especially to allow one symbol or a group as 'argument.'

The need for this specific way of coding comes from the wish that

> the catcodes of the 'parameter' at the time of processing must be different from the (permanently) assigned catcodes when parsed as argument.

Moreover, the catcode changes must be kept local. TeXnically this comes down to the wish that `\foo{<text>}` must be processed as for example

```
\bgroup\x\y\x <text>\egroup
```

with `\x`...`\z` any command, especially catcode changes. This simple case can be achieved by the following code.

```
\def\foo{\bgroup\x\y\z\let\dummy}
```

The paradigm is based on the reading away of the brace preceding `<text>`.

`\if@savingargument` indicates that a parameter will be processed on the fly in `tugboat.sty`. An example of `tugboat.sty` 'processing on the fly coding' is in `\figure...\endfigure`.

```
\figure <pic>\endfigure
%yields as essential replacement text
\topinsert %via\@checkoptions
<pic>      %picture
\endinsert %via \endfigure
```

Another branch of the coding provides for processing the $\langle pic\rangle$ture via

```
\setbox\T@stBox=\vbox\bgroup\hsize\pagewd
<pic>
\egroup
```

[56] The eye-opener how to do this simple was supplied in manmac! Had been there from the beginning.

[57] A white lie. I could not resist to finish it up with a verbatim mode suite of macros. See my BLUe's Verbatim.

[58] Because my btable macro used also `\caption` I had to kludge by providing `\caption{...}` as optional argument within this context. That worked more or less. I decided to change the name into `\capbtbl`.

[59] Within my btable macro I adopted when `\caption` is empty it won't be typeset. Reasonable right?

[60] A modification of LaTeX's footnote in the TeX spirit is provided in TTN2.4 by Jeremy Gibbons. It does not say that verbatims can't be used. No, but when special characters are used in verbatims they will be processed with probably unintended catcodes.

Explanation. As stated earlier `\@checkoptions` also executes to the end of the environment, meaning that it also invokes `\@beginfigure`. The latter codes two possibilities, the branches

- a fixed figure (and on the first page) starts with `\setbox\T@stBox=\vbox\bgroup \hsize\pagewd`, with the material in the input stream—the figure—to follow. On encountering `\endfigure`, `\egroup` is inserted, `\@caption` invoked and the page length adjusted. The latter via `\resetpagelgt`. The testbox is copied into one of the boxes `\firstf@ot` or `\firsth@ad`. The figure will then appear at the top (default), respectively bottom if the option `\bot` has been used.
- a floating figure is handled via the invocation of either `\midinsert`, `\botinsert`, or `\topinsert`, followed by the figure material from the input stream. It ends with `\endfigure`, which after `\@caption` inserts `\endinsert`.

Spanning figures have to be marked up by `\twocolfigure`, which has been defined as

```
\def\twocolfigure{\figure[\fixed]}
```

Apart from the floating aspects of a picture it would be nice if `tugboat.sty` provides for a picture environment à la LaTeX. Why not take the relevant macros from `gkpmac`? If I place myself in the position to answer that question then I would say: 'Some work has to be done to adapt these macros to the general coding philosophy.' Hmmm, not nice! Easy extension should not be hindered by the coding philosophy. I favor a practical and pragmatic extensible system.

2.5 Page make-up, or the OTR.

This is in general an advanced TeX issue and can best be appreciated after chapter 23—and to a lesser degree chapter 15—of the TeXbook have been understood. *TUGboat*'s output routine—OTR for short—is simple in the sense that it just uses `\box`255, also called `\@cclv`. The height of the text in `\@cclv` has been adapted in view of the floats to be added on the page in the OTR, via plain's page breaking mechanism as detailed with in chapter 15 of the TeXbook. To be concrete the amount of space the footnotes require has been accounted for. *TUGboat*'s OTR is complicated in the sense that it allows 1- and 2-column format, and that it allows for overlays to be handled either within TeX or at the dvi-level.[61] The latter means that pages are shipped out which contain the columns separately, to be pasted up later.

Add to this the quote from TeXbook p.253

'. . . Chapter 22 taught you how to be a TeX Master, i.e., a person who can produce complicated tables using `\halign` and `\valign`; the following material will take you all the way to the rank of Grandmaster, i.e., a person who can design output routines. . . . '

and the conclusion will be that the following is for hackers only. Furthermore, *TUGboat*'s OTR is under reconstruction.[62]

Basically, a page can be shipped out via plain's `\plainoutput`, as follows

```
\shipout\vbox{\makeheadline
              \pagebody
              \makefootline}
\advancepageno%TB 252
\ifnum\outputpenalty>-20000 \else
      \dosupereject\fi
%with
\def\makeheadline{\vbox to 0pt
   {\vskip-22.5pt
    \line{\vbox to8.5pt{}\the\headline}
    \vss}
   \nointerlineskip}
%and
\def\makefootline{\baselineskip24pt
    \line{\the\footline}}
```

If we look at the above for the headers and footers and those which actually appear in *TUGboat* then the conclusion is that the coding has been done with more complicated headers envisioned. For the moment plain's simple approach will do, apart from the trim marks.

Plain's `\pagebody` reads

```
\def\pagebody{\vbox to\vsize{%
   \boxmaxdepth\maxdepth\pagecontents}}
%with
\def\pagecontents{\ifvoid\topins\else
   \unvbox\topins\fi\dimen@=\dp255
   \unvbox255 \ifvoid\footins\else
   \vskip\skip\footins
   \footnoterule
   \unvbox\footins\fi
   \ifr@ggedbottom\kern-\dimen@\vfil\fi}
```

In the TeXbook p. 257 it is detailed how to handle two columns. In `tugboat.sty` the columns are set in `\column1`, and `\column2`, and appropriately overprinted on the page.

So, I would suggest in order to make it simpler to provide two separate OTR's: one for overlays to be handled within TeX—then only the switching between 1- and 2-column has to be accounted for—and another to handle the overlays outside of TeX, which ships out each column appropriately.[63] Better still is to provide a separate preprint style with a simplified OTR. For a proposal see Appendix C.

2.6 Back to TUGboat's OTR

Let us forget about the anachronism of the alphatype typesetter and pass through the code to find out what is going on.

[61] Barbara Beeton communicated that the latter is '. . . an artifact of the alphatype typesetter, ... Rather than providing separate output routines in `tugboat.sty`, the proper thing is simply to eliminate this code that is no longer relevant. . . . ' Agreed, wholeheartedly!

[62] Even better as Barbara Beeton communicated '. . . there has been for a year or two an enhanced output routine in test form that permits switching (in the plain version) from two columns to one, and vice versa; this is at the moment rather tricky in use, not documented, and not at all ready to be made public; . . . '

[63] Note added: the latter is an anachronism, and no longer relevant.

TUGboat's OTR is called `\output@`. This routine starts with

```
\@saveorship\midpage{\kern\coloffset
    \pagebody\hfil}
%with essentially
\def\midpage#1{\vbox{%space for runhead
   \vbox to\pagelgt{% runhead (or room for)
      \hbox to\pagewd{#1}%the column + offset
                    % runfoot (or room for)
   }}
```

which comes down to set 'columns'—via `\setbox`—and store these in `\column1`, `\column2` et cetera. `\pagebody`—in reality the invocation of `\pagecontents` in there—yields essentially the contents of `\box255`, as can be read from the above (plain) versions of `\pagebody` and `\pagecontents`.

After that `\output@` continues essentially with

```
...
\ifnum\xcol=\maxcols
   \shipout\hbox{\xcol\@ne
      \loop\rlap{\boxcs{column\number\xcol}
          \ifnum\xcol<\maxcols
              \global\advance\xcol\@ne
      \repeat
      \hbox to\pagewd{}}
\else\global\advance\xcol\@ne
\fi
...
%with from tugboat.cmn
\def\boxcs#1{\box\csname#1\endcsname}
```

to ship out the page or to continue formatting, typesetting and storing the next column(s).

As can be expected `\newpage` has been adapted accordingly to cope with stored 'columns.'

```
\def\newpage{\vfill\eject
   \loop\ifnum\xcol>1
     {\leavevmode\endgraf
      \vfill\eject}% \xcol is
   \repeat}        % advanced in OTR
```

The other problem of handling the last page is generally not an issue because articles may start at the end of the previous one. To assist the editor in this process some macros have been coded for 'page adjustment.' Too specific and too much detail.

Unfortunately, we have to wait a little for THE new OTR to come out, in order to switch happily from 2- into 1-column and vice versa anywhere on the page, for example to set page-wide tables, confidently.

2.7 Conclusion

A beautiful and handy set of tags have been provided for authors writing in TEX about Any-TEX. The coding of the general mechanism for tags is a marvel of TEXing, but a little hard to understand in its full generality.

Much attention has been paid to detail such as the automatic suppression of innocuous spaces and superfluous `\par`-s and blank lines.

I would like to see that `\<foo>*<head text>*` will be used to mark up the head, and `\<foo><body text>\end<foo>` for larger chunks, with the heading set implicitly. The markup for keywords and abstracts can best be done by the latter. The checking of the corresponding `\begin<foo>` and `\end<foo>` can be omitted since the advent of (LA)TEX intelligent editors. Together with the above suggested discipline of use—the environment is prompted by the intelligent editor—this can make the coding of the style (conceptually) much simpler.

Another royal road to simpler coding is to introduce `\this<foo>`-s next to the usual `\every<foo>`-s, instead of 'optional parameters.'

It would be nice to have the handling of verbatims separately available as a tool to cooperate with AnyTEX, especially LATEX.

The two distinct functionalities for the OTR must be (and will be) curtailed with respect to anachronisms. (Actually, the OTR will be replaced in due time.)

The uniform coding philosophy is an advantage with as the negative side that `tugboat.sty` is not easily extensible.

`tugboat.sty` was my style of choice to work with next to manmac. Because of the complex overhead and its incompleteness I will use simpler means to suit my purposes in future.

For example, for the markup of this article I used from `tugboat.sty` the functionalities

- abbreviations (from `.cmn` actually)
- title markup
- `\article` ... `\maketitle` `\endarticle`
- major structure markup commands, like `\head*...*`
- verbatim functionalities in all its glory
- the default `\twocol` (the OTR).

I needed as extra

- `\keywords` ... `\endkeywords`
- `\abstract` ... `\endabstract`
- `\bitem` (bulleted items, and Knuth's `\item`, well I needed them because I like to be independent as much as possible from a specific style)
- `\quote` ... `\endquote`
- (primitive) picture markup
- simple table markup (my `\btable`)
- bibliography handling (my BLUe's Bibliography)
- table of contents
- miscellaneous control sequences, such as `\cs` (to mark up a control sequence), `\\`, `\em` (emphasize), `\partlogo` (from TEXbook), `\ftn` (with automatic numbering).

If I add my own markup commands for title handling, major structure markup commands, verbatim handling, `\multicolumn`—the hardest part, however, I don't need a sophisticated one—then I have the same functionalities, but simpler, faster, and more flexible. This emerged eventually in my proposal `tug.ppt`.

ltugboat.sty

3 LTUGBOAT.STY

For LaTeX submissions the TeX Users Group relies on LaTeX's `article.sty`, and `art10.sty`. In the users' guide only one column is devoted to the use of `ltugboat.sty`. The version I inspected is 1.16e, Dec 92. Because this article is not about LaTeX adaptation this part is brief.

There is a problem in customing here. On the one hand we have the LaTeX parameters which can be adjusted and on the other hand we have `\hsize` and `\vsize` which are used by TeX now and then automatically at a lower level.

From the LaTeX file—Output section—the parameters involved are given below.

- `\textheight`, height of text on page, excluding head and foot
- `\textwidth`, width of printing on page
- `\columnwidth`, in one-column the value `\textwidth`, and in two-column the value (`\textwidth-\columnsep`)/2.

Also `\@colht` and `\@colroom` must be initialized by the value of `\textheight`.

3.1 Customing

In order to let the appearance in print be similar to plain TeX submissions some parameter settings have been altered.

The page size parameters involved are enumerated below.

```
\textheight=54pc
\columnwidth=18.75pc
\columnsep=1.5pc
\textwidth=39pc
%2\columnwidth+\columnsep=\textwidth
```

Headers and footers. Peculiar is

```
\def\PrelimDraftfooter{%
 \dlap{\kern\textheight\kern3pc\rlap{\hbox
  to\pagewd{\midrtitle\hfil\midrtitle}}}}
```

This 'footer' is invoked in

```
\@oddhead
```

along with the header material, while `\@oddfoot` has been left empty. Why? Weird![64] Similar things hold for the 'even' counterparts.

Sectioning commands. The LaTeX commands are redefined with as replacement text the invocation of `\@startsection` supplied with the appropriate argument values, to yield a `tugboat.sty`-like result. For example

```
\def\section{\@startsection{section}%
    {1}{\z@}{-8pt}{4pt}%
    {\normalsize\bf\raggedright}}
```

It is also stated

> 'Redefine style of section headings to look more like *TUGboat*. Start with redefinitions from `article.sty`. (Only `\section` correct so far.)'

Funny looks

```
\def\abstract{\section{Abstract}}
\def\endabstract{}
```

It is a bit against the philosophy of `\<foo>...\end<foo>`, which in `tugboat.sty` delimit the header, and for this case 'Abstract.' Now the word abstract has been wired in and has made this command language specific, and moreover `\<foo>...\end<foo>` should now enclose the *contents* of the abstract. (But because `\endabstract` is empty it can be placed anywhere?!?) This cosmetics is confusing. I will just stay with LaTeX's

```
\section*{Abstract}
<Abstract text>
```

or redefine it with the original and general function, with use

```
\abstract
<Contents abstract>
\endabstract
```

As can be seen from this I'm not at all against this markup, but the code looked just so strange.

Footnotes. The only thing adapted is the appearance of the actual footnote via

```
\long\def\@makefntext#1{\parindent1em
   \noindent\hbox to2em\llap{\@makefnmark}
   \null$\mskip5mu$#1}
```

As stated before the functionalities differ, especially LaTeX does not allow verbatims in footnotes (fragile!).[65]

Lists. Adapted are the default values of LaTeX's lists as required by LaTeX in `\@listi`, `\@listii`, and `\@listiii`.

For example

```
\def\@listi{\leftmargin\leftmargini
 \parsep=1pt plus 1pt minus 1pt
 \itemsep\parsep
 \listparindent=1em}
```

The latter macros are invoked in LaTeX's `\eval...`. For customing LaTeX's lists see Goossens, Mittelbach and Samarin's paper on the issue.

Verbatims. Nothing is adapted here. So as long as LaTeX does not provide file verbatim this is not supported either in `ltugboat.sty`.[66]

Figures. Here the title and number of the caption are customized via

[64] Barbara Beeton communicated '... It happens that there is a good reason for this. Placing both header and footer in the 'header' with a fixed distance between them ensures that this distance will never change. If the regular footer was used and the page overfull, the footer would move down by that overfull distance, sometimes even disappearing off the page. It is easier to diagnose an overprinted footer than an absent one, and the fact that a bottom-of-page element can be relied on always to be in exactly the same location provides an extra check on the general page setting, without having to look at the job log. ... ' Hmmm, it seems a kludge to me, because of a deficient OTR. Nothing wrong with kludges, but mark them as such.

[65] Note that the parameter is superfluous—like my late `\ftn`—it is just reinserted at the end.

[66] However, a file verbatim style is available. See `\verbatiminput` in Schöpf, 1989.

```
\def\fnum@figure{{\bf Figure\thefigure}}
```

Bibliography. Essentially a section with (hardwired) title 'References' has been defined and the references that follow are formatted via (LaTeX's) `\list`, as shown by the code below.

```
\def\thebibliography#1{%
 \section{References}
 \@mkboth{REFERENCES}{REFERENCES}}%
 \list{[\arabic{enumi}]}{\settowidth
  \labelwidth{[#1]}\leftmargin\labelwidth
  \advance\leftmargin\labelsep
  \usecounter{enumi}}
 \def\newblock{\hskip.11em plus.33em
              minus.07em}%\sloppy
 \tolerance8000\hfuzz.5\p@\vfuzz.5\p@
 \clubpenalty4000\widowpenalty4000
 \sfcode‘\.=1000\relax}
```

3.2 Conclusion

It is amazing that LaTeX's `article.sty` could be adapted to mimic the appearance of `tugboat.sty` by so few commands. It is a pity that some necessary functionalities provided by `tugboat.sty` are not available in `ltugboat.sty`—like file verbatim—to guarantee that published macros are exactly those which have been used. With respect to non-English use the hardwired-ness of several English titles is unhandy.[67]

LaTeX provides the command `\pagestyle` for page layout mods. This could have been used with let us say *tugboat* page style to be defined via `\ps@tugboat`. Why not? An example of how to do this is provided in `eurotex.sty`.

Barbara Beeton communicated that one of her priorities with respect to adaptation of LaTeX is to remove '. . . the inability to suppress the indent on the first paragraph after `\maketitle`.'

tugproc.sty

4 TUGPROC.STY

This customing of `tugboat.sty` is only some 350 lines. I inspected version 1.10, June 92.

The 'Guidelines for the proceedings' are nicely done. Useful are the inclusion of spelling conventions and information about font use. I inspected Guidelines version 1.06, May 93.

4.1 Customing

The adaptations concern mainly the title part and the back matter. For the title part the title is set left justified, followed by the complete author information, also left justified. Keywords are discouraged. The abstract is centered and spans two columns. No section numbering. After this the copy proper is set in two-column format as basis. The appendices start on a new page and are set in one-column.

The page size parameters involved have not been altered.

The headers and footers have a r(unning) header and footer. The title page of each article does not have a running head. The other pages contain on the even numbered pages the title and on the odd numbered pages the author name(s). The footers contain the issue information and the conference information (inbound), next to the page number (outbound).

`\article` takes care of the formatting of the title and abstract,[68] via essentially

```
\def\article{\setbox\startbox=
   \hbox to\colwd{
     \hbox to\pagewd{\vbox{%title}}
     \hbox to\pagewd{\hfil\vbox{
         %abstract head and material
         }\hfil}\vskip1pc}}\hss}
%
\twocolfigure\box\startbox\endfigure
}
```

Abstract. I like this `\abstract <abstract contents> \endabstract`. Simple and straight. Note, however, how the coding has been adapted to allow for `<abstract contents>` spanning the two columns. The command `\article` invokes `\theabstract`. The latter has been defined in `\endabstract` as can be seen from the definitions copied below.

```
\def\abstract{\@abstract[\longargument]}
\def\@abstract{\begingroup
   \def\CurrentTag{abstract}
   \@defaultoptions
   \@savingargumenttrue
   \@checkoptions}
%
\def\endabstract{\global\toks@=
   \ea{\the\@argument}
   \endgroup
   \edef\theabstract{\ignorespaces
      \the\toks@\unskip}}
```

Bibliography sets entries via

```
\head*Bibliography*
\entry{Laan C.G van der,
  ‘‘BLUes Bibliography’’,
  MAPS 93.1, pages 205--210.}
%et cetera, with
\def\entry#1{\noindent\frenchspacing
    \hangindent\Hang#1}
\def\Hang{1em}
```

Appendices start on a new page in one-column format. The title of the appendix is set via

```
\def\appendix#1\endappendix{\newpage\onecol
\centerline{\HEADfont#1}}
```

That this coding does not allow for the `*`-delimiters indicate that the editors did not want to adapt—or did not feel comfortable with—the general mechanisms available in `tugboat.sty`. I for one find the mnemonics `\endappendix` misleading. (With the regular end tags it does not hinder because then the `*`-s can be used.) It suggests the end of the appendix. From the 'Guidelines' the following with respect to the use of it.

[67] Agreed, *TUGboat* is in English.

[68] Note that therefore this command must follow the abstract in contrast with tugboat.sty's `\article`.

```
\appendix Appendix\endappendix
\head*Spelling Conventions*
<Contents>
```

bf Coding: alternative. The general mechanism could have been applied, with automatically the flexibility with respect to the use of `*`-separators and the options which come with any head. (Similar to coding of `\head`.) However, I chose to keep it simple and to favor a name for the word appendix `\nameappendix`.

```
\def\nameappendix{Appendix}
\def\appendix{\newpage\onecol
   \centerline{\HEADfont\nameappendix}
   \DeleteOptionalSpacesandPars
      {\noindent\ignorespaces}}
```

A user can customize `\nameappendix`, that is all. The user reads

```
\appendix
\head*Spelling Conventions*
<Contents>
```

This use of `\appendix` is different from the use of `\keywords`, `\head` and the like. It has a different function too.

4.2 The modification of the OTR?

Because the appearence of the page is different from regular issues of *TUGboat*—especially the first page—we expect that the OTR has been modified. In reality the structural modifications have been done via `\article`. The modification in `\midpage` is not essential. (The footnotes are overprinted, similar to the way it has been done in `ltugboat.sty`.)

4.3 Conclusion

The layout of the front matter and the back matter have been adapted. Also the page layout has been customized in agreement with the purpose.

Editorial boards can learn from this file how to customize `tugboat.sty`. However, the file also witnesses that it is difficult to remain consistent with the coding conventions already present.

Why not provide a `tug.ppt` style and shield authors from the differences?

ltugproc.sty

5 LTUGPROC.STY

This customing of `ltugboat.sty` is only some 200 lines. I inspected version 1.06a, Jan 93. Basically, the title runs over two-columns, the (complete) author information is left justified, and the abstract spans also two-columns and is part of the title. It suppresses section numbering (`\setcounter{secnumdepth}{0}`). and adheres to different parameter settings for white space around the headings.

The authors' guide is 'Guidelines for the proceedings,' the same as the one with `tugproc.sty`.

5.1 Customing

The page size is left invariant in correspondence with *TUGboat*.

The headers and footers have their own layout and contents via `\@oddhead`, `\@oddfoot` and there 'even' analogues. Customing is as follows, where the account for the suppression at the title page has been omitted

```
\def\@oddhead{\hfil\rm\rhTitle}
\def\@evenhead{\rm\rhAuthor\hfil}
\def\@oddfoot{\issue\hfil\thepage}
\def\@evenfoot{\thepage\hfil\issue}
\def\issue{MAPS 94.1}
```

The `\dopagecommands` is technical for the editors, I presume.

Section heads. Remarkable is the attention given to the markup of the abstract. A minipage centered after the title. `\abstract` provides the word 'Abstract' appropriately set, and starts a minipage and within that a list. `\endabstract` terminates the list and the minipage.

Appendices. To have appendices in one-column a `\onecolumn` command has to be inserted before each `\section{Appendix: ...}`.

Bibliography. In contrast with `ltugboat.sty` the LaTeX convention has been abstracted from.

```
\section{Bibliography}
\bibentry <name>, ''<title>''
   <journal>, <pages>, <year>.
%et cetera
```

5.2 Switching into tugboat.sty

To get an idea of the differences in markup and the amount of work involved, I transformed the markup of my BLUe's Bibliography paper from `ltugproc.sty` into `tugboat.sty`.

The markup which had to be changed is supplied in the accompanying table.

ltugproc.sty copy	⟶	**tugboat.sty copy**
`\documentstyle`{ltugproc}	⟶	`\input` tugboat.sty
`\begin`{document}		
`(ltug) title markup`	⟶	`(TB) title markup`
`\begin`{abstract}	⟶	`\abstract`
`\end`{abstract}	⟶	`\endabstract`
`\maketitle`	⟶	`\article` %Before abstract
%No keywords	⟶	`\keywords` ...`\endkeywords`
`\section*`{...}	⟶	`\head*`...*
`\subsection*`{...}	⟶	`\subhead*`...*
`\begin`{itemize}	⟶	`\␣`
`\end`{itemize}	⟶	`\par`
`\item`	⟶	`\bitem` %(essent.plain's)
`\begin`{quote}	⟶	`\quote`
`\end`{quote}	⟶	`\endquote`
`\footnote`	⟶	`\ftn` %with aut.numbering
`\begin`{verbatim}	⟶	`\verbatim` %(BLUe's Verb.)
`\end`{verbatim}	⟶	!endverbatim
`\vrb|`	⟶	`|`
`\ldots`	⟶	`\dots`
`bibliography markup`	⟶	. . . %(BLUe's Bibl.)
`\end`{document}	⟶	`\makesignature`
	⟶	`\endarticle`

Near all is reversible, except for the itemize part. It did take me an hour to convert this 6 page—2-column—article. Just change the style file? Forget it. The way out for authors is a common `tug.ppt` (preprint) style, to abstract from the regular and proceedings *TUGboat* issues.

5.3 Conclusion

An annoyance is that when using `ltugproc.sty` I have to remember that in the markup the `\maketitle` command must be supplied *after* the abstract, and not before as usual with LaTeX. This is because of the choice to have the abstract span two columns. (`\@maketitle` has been adapted.) I also have to remember that LaTeX's bibliography environment is not used. Too much detail to remember, so I will rely on my example template—empty article from last time—when I need `ltugproc.sty` again.

`euro92.sty`

6 EURO92.STY option

This style option has been designed by Rick Furuta in 1987, and used and adapted by Philippe Louarn for EuroTeX '91, and by Petr Sojka for EuroTeX '92.

Some may ask what does this have to do with TUG? Frankly, there is no direct link when looked at it from a formal point of view. However, I consider this also a descendant from the *TUGboat* styles, and more importantly it shows how to use LaTeX's `\ps...`, to customize for page styles. Next to that is my wish to have the styles I have to deal with discussed in one context.

The option is an adaptation to LaTeX's `article.sty` and some 290 lines long. `\@maketitle` has been recoded to account for the various white spaces, the `\hrule`, and the fonts used. The headings are customized with respect to the contents of the arguments and also to account for the settings of the surrounding white space. There is no version indication included nor a history of changes.

The 'Guidelines for authors' has been done nicely. LaTeX authors can just start from the template—the guide itself—and obey the supplied information. TeX authors are requested to obey the included specifications. Good! Perhaps it is too much to ask from a TeX author to obey the specifications. The temptation to ride one's hobby in details is always there.

6.1 What does the EuroTeX proceedings look like?

Basically a table of contents with the sequence of articles next. It is essentially one-column biased. In each article the title is underlined by a `\hrule`. The author information is set left justified, followed by the (centered) abstract and (centered) keywords (in one or two languages). Then the sections follow. Appendices are at the end. The headers on the pages have the author name (in italics and underlined with a page-wide `\hrule`) on the left-hand pages, and the title of the article (in italics and underlined with a page-wide `\hrule`) on the right-hand pages. As exception the first page of each article has the information about the conference (in roman and underlined again with a page-wide `\hrule`). The footers contain the page numbers outbound.

The typographic quality of the proceedings I have seen is moderate, with all respect. Trivial errors in there. English proof readers could have contributed too.

6.2 Customing

The page size parameters are as follows

```
\textheight=546pt
\textwidth=12.7cm
%and some others involving \parskip and
%various separators
```

Headers and footers are controlled by

- `\def\ps@titre`, for title page style (with the conference information appropriately formatted)
- `\def\ps@gut`, with `\@oddhead`, `\@oddfoot`, and their even analogues, appropriately defined.

The author has to provide the information for the running heads in

```
\titlehead{<short title for paper>}
\paperhead{<author name(s)>}
```

At the end of the style file `\pagestyle}{gut}` and `\thispagestyle{titre}` take care that the appropriate information will be used.

The LaTeX macro `\@outputpage` has been modified to include the underlining of the headers via the insertion of

```
\vskip10pt
\hbox to\textwidth{\hrulefill}
```

For the markup of other material just use LaTeX.

Intermezzo: abstraction from def versus toks variable. At the user level one can't tell from for example

```
\titlehead{<Short title for paper>}
```

whether this is implemented as a toks variable or a (second level) definition.

```
\newtoks\titlehead
%or, as is the actual case
\def\titlehead#1{\gdef\@titlehead{#1}}
```

The advantage of introducing the second level defs is that the user doesn't have to think of the `\def` token. For a novice there is confusion.
End intermezzo.

6.3 Conclusion

This style shows how LaTeX's OTR—or more specifically `\@outputpage`—can be customized. The Guidelines for authors is in itself a nice template for an author's article. Adaptation via `\ps@...` is well-done, and that is the way how it must be done.

The bad news is—I'm sorry to say so—that the resulting typographical quality has been moderate.

ttnxnx.sty

7 TTNXNX.STY

This short style for TTN is based on LaTeX's report, and essentially 1-column, apart from the index. It contains its own abbreviations.[69] It checks whether the NFSS is in use and if so it takes appropriate action. Each major part has its title centered enclosed by `\hrule`s. Sections have the titles left and author information flushed right. I inspected `ttn2n3.sty` of 1993.

For submissions of NTG's meeting reports I could just submit the ASCII copy, and the editor—Christina Thiele—inserted the few markup commands en-passant, while polishing my use of English.

7.1 Customing

The page size parameters involved are

```
\textwidth=29pc
\textheight=43pc
\voffset=-2pc
\overfullrule=0pt
\hfuzz=5pt
```

The headers and footers have a r(unning)-title and from the style I can't find how they were included.[70]

Title parts are redefined as follows

```
\newcommand{\Section}[1]{\section*\centering
   \hrule\hrule\vskip.5pc{\Sectionfont #1}
   \vskip.5pc\hrule\hrule\vskip1pc}
\squasheadsubsection{foo}{bar}
%gives similar results to
%\subsection*{foo}\vspace*{-1.5pc}
%\begin{flushright} bar\end{flushright}
%\noindent
```

Too much detail and too specific to discuss the code here.

7.2 Conclusion

The newsletter is and looks great. Again, to my knowledge, no formal specifications are there for the layout of TTN. They just grew.[71]

— — * — —

8 Looking back

The style files are a rich source and the documentation is well-done. The coding of `tugboat.sty` is superb, although unnecessary complicated now and then. Nevertheless, I learned a lot from them. But this monolithic way of coding is not my style. I prefer orthogonality and modularity.

Given that many people have been involved at geographically widely spread places—and worked under the pressure of continuous lack of time—I'm happy to conclude that the amount of inconsistencies is low, very low, and that the overall quality is good, although a bit difficult to read. But that is in the nature of TeX being so unusual, I presume.

It is useful to take over the abundant good ideas which popped up at many places in the styles. As a non-TeX specific example one can think of the 'spelling conventions' useful for non-native speakers of English. These should be made separately available to grow and to be generally used.

In working on this article I realized more and more the specific function of keywords and abstract, next to other bibliographic information. Because of this it is worthwhile to typeset these differently—I chose a smaller type—from the copy proper.

For optional parameter handling as done in `tugboat.sty` a much simpler alternative is to make use of `\this`<foo>, analogous to `\every`<foo>.[72]

If we compare the `amsppt.sty` with `tugboat.sty` then I conclude that the handling of optional parameters goes much along the same lines. Furthermore, `amsppt.sty` is more developed with respect to the formatting of math, and `tugboat.sty` more with respect to handling verbatims.

Styles with a rigid and advanced coding approach tend to become cost-intensive to maintain, especially with adaptation

- to changing circumstances—the example in the past has been the (user) migration from (flexible) plain into (rigid) LaTeX, the latter has been under revision for the last 5 years—or
- to extending it with useful functionalities developed elsewhere under a different coding philosophy.

This phenomenon is general.

> In the past the software engineers have experienced similar difficulties with big and rich computer languages—e.g. ALGOL 68, ADA, . . . , and not to forget huge operating system (as opposed to the RISC approach)—which have given birth to the phenomenon called 'little languages.'

[69] This demonstrates the need for a separate file for abbreviations to be used within any context.

[70] Christina communicated '. . . They're inserted into the source `.tex` file, not in the `.sty` file.

[71] Christina Thiele communicated 'The same, more or less, can be said about the *TUGboat* and proceedings macros: there has *never* been enough time to make sure that the macros are all done nicely, to agree with philosophical and overall consistency issues. They just grew, because over time, different situations arise, and it's more important to solve the immediate problem than to go back through the entire set of macros, to make sure that everything is still consistent and looks nice.' Fair enough. So be it. Like monolithic coding this *continuously* working 'behind the time' is not my style either. If I would experience *continuously* working behind the time, bells would ring that something is wrong there, it should be rearranged and in general I would face the choice '*dare to do less,*' as communicated at the Aston BoD meeting.

[72] This applies also to amsppt's style.

There are always quibbles from outsiders like me. I hope nevertheless that my remarks are well-taken and/or will assist people in understanding what is going on. The very least is that my reflections show how the styles are perceived by BLU.

The TEXing paradigms I stumbled upon are summarized below.

- data abstraction—or hiding the `\def` token[73]
- alphanumeric numbering
- optional parameter handling
- outer def-s as argument of a test by hiding the test in another layer
- allow for either *-separators or an end delimiter
- 'parameter separator' with optional spaces before and after, via parsing
- minimal user markup for verbatim, next to complete `\verbatim`...`\endverbatim`
- flexible escape mechanism in verbatim
- end verbatims via |<escape character>egroup|
- parameterization over parameter delimiters
- processing on the fly of 'arguments.'

I introduced myself the concept of `\this<foo>`.

9 Looking forward

Software engineering has it that for complex projects the concept of proto-typing is used. A realization of this is that when software is designed and coded it will be thrown away, and with the knowledge gained during the project new specs and code will be developed. To quote from Heckel

'Prototype, revise, and rewrite.'

Pondering about the above in relation with the TUG style files, the following came to mind.

I would first like to see the specs—purpose, specifications, examples of use—and a set of requirements to which the code should obey. Then I would ask myself the (rhetorical) question whether the *TUGboat* styles should phoenix.[74] I mean redesigned with simplicity, flexibility, generality, extensibilty and robustness as yard sticks, in the form of a literate program. This entails that the development of the code goes hand in hand with the emerge of the documentation. The extra bonus of this approach is the immediate availability of an index and the change file concept as tool for customing.

Perhaps the above idea is already outdated in view of the hypertext developments. What about 'Style files as an hypertext,' to paraphrase Mary Dyson?

Acknowledgments

Barbara Beeton helped me with recent copies of the style files and with some comments as well as with her lecture at the SGML-TEX meeting, 1990, at Groningen. Next to that she proofed the article and pointed out several misconceptions, and made suggestions for improvement. Christina Thiele polished again my English.

Włodek Bzyl contributed to a clearer description of the 'generic coding pitfall,' next to clarifications of the used terminology, and pointing to the `gkpmac` format used for typesetting 'Concrete Mathematics.' Moreover, when I talked to him about my ideas to continue my BLUes trilogy with *TUGboat* BLUes, he pushed me forward a bit, and also challenged me to embark upon literate programming.

Ron Whitney read it nearly all through and put things within the right historical context, next to pointing to some vagueness now and then, especially in the `tugboat.sty` part.

Thank you! Any inaccuracies left are oversights on my part. I welcome constructive comments. History has it that 'flames' are the valuable ones, although wrongly packed. Comments which point to my abuses of English—especially the blind spots—are welcomed in particular.

Bibliography

[1] Beeton B.N (1990): *TUGboat* production: TEX, LATEX, and paste-up. SGML-TEX meeting, Groningen. (Unpublished)

[2] Bzyl W, T Przechlewski (1993): An application of literate programming: creating a format for the Bulletin of the Polish TUG. TUG '93. *TUGboat* 14, no. (3), 296–299. (Very promising this using of the experience embodied in `tugboat.sty` via transformation into WEB and modify this with GUST's mods via change files. As a bonus there is an index. Handy to locate commands, but it won't help to locate the various `\@begin`<foo>-s commands because these are invoked via csnames.)

[3] Dyson M.C (1992): The curriculum as a hypertext. EP-ODD, 5, 2, 63–72.

[4] Gibbons J (1993): Footnotes with verbatim material. TTN, 2, 4, p.9. (LATEX's footnote implementation has been modified to allow also for verbatims similar to plain's coding. Basically it comes down to handling the 'argument on the fly.' Really nice.)

[5] Goossens M, F Mittelbach, A Samarin (1993): The LATEX-companion. Addison-Wesley.

[6] Goossens M, F Mittelbach, A Samarin (1993): customing LATEX's lists. MAPS 93.2, 177–183.

[7] Hamilton H (1989): Mastering TEX with templates. TUG '89. *TUGboat* 10, no. (4), 541–548.

[8] Heckel P (1982): The elements of friendly software design. Warner Books. ISBN 0-446-38040-7.

[9] Knuth D.E (1984): Computers and Typesetting. The TEXbook. Addison-Wesley. ISBN 0-201-13447-0 (hard cover) ISBN 0-201-13448-9 (soft cover). (For the correct printing look in the index for `\language` or `\emergystretch`.)

[73] From `\foo`<copy> a user can't tell whether a toks variable has been used or a (second level) `\def`.

[74] Ralph Youngen from AMS communicated to me at Aston that AMS is thinking about abandoning the `\nofrills` methodology alltogether.

[10] Knuth D.E, S Levy (1987): The CWEB System of Structured Documentation.
(FTP: labrea.stanford.edu, in directory /pub/cweb)

[11] Laan C.G van der (1991): Math into BLUes. Part I: Mourning. TUG '91. *TUGboat* 12, no. (3), 485–501. Part II: Sing your song. EuroTEX '91. GUTenberg Cahiers 10&11, 147–170. (Pre-release MAPS 91.1.)

[12] Laan C.G van der (1992): Tower of Hanoi, revisited. *TUGboat* 13, no. (1), 91–94. Also MAPS 92.1.

[13] Laan C.G van der (1992): Table Diversions, MAPS 92.2, 115–129. (An earlier version at EuroTEX '92.)

[14] Laan C.G van der (1992): FIFO and LIFO sing the BLUes. *TUGboat* 14, no. (1), 54–59. (An earlier version at EuroTEX '92, 225–234, and MAPS 92.2.)

[15] Laan C.G van der (1992): Syntactic Sugar. TUG '93. *TUGboat* 14, no. (3), 310–318. (Earlier versions in MAPS 92.2, abridged in GUST bulletin 1.)

[16] Laan C.G van der (1993): Manmac BLUes—or how to typeset a book via TEX. MAPS 93.1, 171–191.

[17] Laan C.G van der (1993): AMS BLUes—professionals at work. MAPS 93.1, 192–212.

[18] Laan C.G van der (1993): Sorting in BLUe. MAPS 93.1, 149–170. (Abridged TUG '93. *TUGboat* 14, no. (3), 319–328.)

[19] Laan C.G van der (1993): BLUe's Bibliography—a generic approach. MAPS 93.2, 205–210.

[20] Laan C.G van der (1994): BLUe's Transparencies—From report to transparency. MAPS 94.1, 111–114.

[21] Laan C.G van der (1994): BLUe's Verbatim—The selection MAPS 94.1, 116–118.

[22] Lamport L (1985): LATEX User's Guide & Reference Manual. Addison-Wesley. ISBN-0-201-15790-X. (With respect to BibTEX the following characteristics from UKTUG meeting of 1990. Advantages: clear layout of database, unique identifyer for elements, compatibility with Scribe databases, explicit statement of citation type, extensible style language, uses LATEX's cross-referencing, easy to edit output, can be mixed with non-automatic generated bibliographies, cross-referencing and abbreviations. Disadvantages: multiple passes (LATEX, BibTEX, LATEX, LATEX); have to remember unique references, style language is opaque, database is very wordy and boring to enter.)

[23] Louarn P (1991): Instructions to EuroTEX and GUTenberg '91 authors.

[24] Mittelbach F (1989): An environment for multi-column output. *TUGboat* 10, no. (3), 407–415.

[25] Mittelbach F (1989): The doc-option. *TUGboat* 10, no. (2), 245–273.

[26] Mittelbach F, R Schöpf (1989): The new font selection scheme. Reprint: *TUGboat* 11, no. (2), 297–305. (See also NFSS 2, in Goossens, Mittelbach Samarin.)

[27] Mittelbach F, C.A Rowley (1993): The LATEX3 project. MAPS 93.1, 95–99. (Also in various early nineties proceedings.)

[28] Rahtz S.P.Q, M Burbank (1993): Guidelines for Proceedings of the 1993 Annual Meeting of the TEX Users Group. TUG Office.

[29] Salomon D (1992): NTG's Advanced TEX course: Insights and Hindsights. MAPS Special, $\approx 500p.$

[30] Schöpf R (1989): A new implementation of the LATEX `verbatim[*]` environments. *TUGboat* 11, no. (2), 284–296.

[31] Spivak M.D (1989): L$\mathcal{AMS}$-TEX—The Synthesis. TEXplorators Corporation.

[32] TUG (publications): *TUGboat*, TTN, TEXniques.
(*TUGboat*: Scholarly quarterly of TUG, with the proceedings of the annual meeting as special issue, and the resource directory as supplementary issue. From 1980 onward. The style files involved are `tugboat.cmn`, `tugboat.sty`, and `ltugboat.sty` available from the (CTAN) archives.
TEXniques: Special themed editions, with no 1 to 14 of 1993) TTN: A portable quarterly newsletter of TUG. From 1990 onward.)

[33] Whitney R.F, B.N Beeton (1989): *TUGboat* authors' guide. *TUGboat* 10, no. (3), 378–385. (Updated versions via the file server.)

— — ∗ — —

Appendix A: Templates

In the templates I incorporated a simple and characteristic way of markup for the title part, sections, footnotes, lists, verbatims, figures, bibliography and appendices. From this a reader can easily 'jump off' from the template in the spirit of the work of Hope Hamilton.

The anthology shows that the markup is a little varied.

It is hoped in making the diverse markups explicit developers will realize that nearly the same functionality can be reached with less, much less variation. Syntactic Sugar? Yes!

A.1 Template: `tugboat.tem`

```
%tugboat.tem for .sty version 1.14
\input tugboat.sty    %\input tugboat.cus
%
\def\runtit{TUGboat BLUes}
\title*\runtit---{\rm how \TeX ies do it}*
\author*Kees van der Laan*
\address*Hunzeweg 57, 9893PB, \dots*
\netaddress*cgl@risc1.rug.nl*
%
\article
%\head*{Keywords}* I use the mod
\keywords tugboat.sty, \dots
\endkeywords
%\head*{Abstract}* I use the mod
\abstract Abstracts and keywords
   are special. I use my variants.
\endabstract
%
\head*Introduction*
Blah, blah and more blah \dots
\head*Sectioning commands*
Nothing special, just |\head*...*|,
|\subhead*...*| and |\subsubhead*...*|,
to be followed by the contents of the
sections.
```

```
%
\head*Footnotes*
Text with verbatim in
footnote.\footnote*{|Verbatim| text.}
%
\head*Lists*
\list
\item Note |\item| has been redefined
   within |\list...\endlist|.
\endlist
%
\head*Verbatims*
I love |\foo{<argument>}|, possible via
|\enablemetacode| option.
%
%Tables? Generic, via my \btable?
%Math? See the TeXbook
%Symbolic cross-referencing?
%  (See Spivak or my Math into BLUes)
%
\head*Figures*
Floating objects via
||\fig<vertical material>\endfig||
%
\head*Bibliography*
Not supported other than plain’s
facilities. I use
\item{[1.]} Laan C.G van der (1993):
   BLUes Bibliography. MAPS 93.2, 205--210.

\head*Appendix A: Templates*
Appendix, nothing special.
\makesignature
\endarticle             %cgl@risc1.rug.nl
```

A.2 Template: `ltugboat.tem`

```
%ltugboat.tem for .sty version 1.16e
\documentstyle{ltugboat}
\begin{document}    %\input{ltugboat.cus}
\def\runtit{Sorting in BLUe}
\title{\runtit}
\author{Kees van der Laan}
\address{Hunzeweg 57, 9893PB,
        Garnwerd, The Netherlands,
        05941--1525; cgl@risc1.rug.nl.}
\date{}
\maketitle
%
\subsection*{Keywords:}Sorting, \ldots
%
\section*{Abstract}
‘Abstract’ according to language.
%
\section*{Introduction}
%Copy proper a la LaTeX
%
Text\footnote{No verbatims!}
%
\begin{itemize}
\item ...
\end{itemize}
%
%Tables? Generic via my \btable?
%Math? See the TeXbook
%
\begin{figure}
\hbox{...}
\caption{...}
\end{figure}
%Back matter
\begin{thebibliography}{abc}
\frenchspacing
\bibitem{cgl}Laan C.G van der (1993):
Sorting in BLUe. \tubissue{14}(3),
319--328.
(Unabridged MAPS 93.1, 149--170.)
\end{thebibliography}
%
\appendix
\section{Heap sort}
\makesignature
\end{document}          %cgl@risc1.rug.nl
```

A.3 Template: `tugproc.tem`

```
%tugproc.tem for .sty version 1.10
\input tugproc.sty    %\input tugproc.cus
%
\def\runtit{Manmac BLUes}
\title*\runtit---{\rm
   or how to typeset a book via \TeX}*
\author*Kees van der Laan*
\address*Hunzeweg 57, 9893PB, \dots*
\netaddress*cgl@risc1.rug.nl*
%
\abstract The manmac macros are \dots
\endabstract
%
\article%Sets title and abstract
%\head*Keywords*CAT, ...%Discouraged
%
\head*Introduction* Nothing special here.
%
\head*Sectioning commands*
Just |\head*...*|, |\subhead*...*| and
|\subsubhead*...*|, and contents.
%
\head*Footnotes*
Text with verbatim in
footnote.\footnote*{|Verbatim| text.}
%
\head*Lists*
\list
\item Note |\item| has been redefined
   within |\list...\endlist|.
\endlist
%
\head*Verbatims*
I love |\foo{<argument>}|, possible via
|\enablemetacode| option.
%
%Tables? Generic via my \btable?
%Math? See the TeXbook
%Symbolic cross-referencing?
%   (See Spivak or my Math into BLUes.)
%
\head*Figures*
Floating objects via
||\fig<vertical material>\endfig||
%
\head*Bibliography*
\entry{Laan C.G van der (1992):
Table Diversions.
Proceedings Euro\TeX ’92, 191--211.
(Adapted MAPS 92.2, 115--129.)}
%
\appendix Appendices\endappendix
\head*A: The file manmac.tex*
Page wide text.
%No signature!
\endarticle             %cgl@risc1.rug.nl
```

A.4 Template: `ltugproc.tem`

```
%ltugproc.tem  for .sty version 1.06a
\documentstyle{ltugproc}
\begin{document}   %\input{ltugproc.cus}
%
\def\runtit{Sorting in BLUe}
```

```
\title{\runtit}
\author{Kees van der Laan}
\address{Hunzeweg 57, 9893PB, \ldots}
\netaddress[\network{%
   Internet}]{cgl@risc1.rug.nl.}
\begin{abstract}
'Abstract' according to language.
\end{abstract}
\maketitle
%
\section*{Introduction}
%Copy proper a la LaTeX
%
Text\footnote{No verbatims!}
%
\begin{itemize}
\item ...
\end{itemize}
%
%Tables? Generic via \btable
%Math? See the TeXbook
%
\begin{figure}
\hbox{...}
\caption{...}
\end{figure}
%Back matter
\begin{thebibliography}{abcdef}
\frenchspacing
\bibitem{cgl}Laan C.G van der (1993):
Sorting in BLUe. \tubissue{14}(3),
319--328.
(Unabridged MAPS 93.1, 149--170.)
\end{thebibliography}
%
\appendix
\section{Heap sort}
\end{document}           %cgl@risc1.rug.nl
```

A.5 Template: `eurotex.tem`

Below the author name must be supplied twice because of the definition

```
\def\authorhead#1{\gdef\@authorhead{#1}}
```

If a toks variable was used instead then we had a @-command less and we could use the toks variable in `\title` too, as follows.

```
\authorhead={<author name>}%= optional
\title{\the\authorhead}
```

The reason I can think of is the *over*use of @-commands. In this case the @-command is completely superfluous. In general it is not clear to me either when to use toks variables and when to use defs. One case where it is extremely clear is when too many toks variables are needed (≥ 256).

```
%eurotex.tem for euro92.sty
\documentstyle[euro92]{article}
\begin{document}      %\input{eurotex.cus}
%
\title{Table Diversions\thanks{A little
   different from proceedings Euro\TeX\ '92,
   especially in the coding of FIFO.}}
 \titlehead{Table Diversions}
\author{Kees van der Laan}
 \authorhead{Kees van der Laan}
\affiliation{Hunzeweg 57, 9893PB,
        Garnwerd, The Netherlands,
        05941--1525; cgl@risc1.rug.nl.}
\maketitle
%
\begin{abstract}Characteristics \ldots
\end{abstract}
\begin{keywords}(Bordered) Tables, ...
\end{keywords}
%
\section*{Introduction}
%
%For other material a la \LaTeX.
%
\begin{thebibliography}{abc}
\bibitem{cgl} {Laan C.G van der}
   (1992): Table diversions.
   Proceedings Euro\TeX{} '92, 191--211.
   (Adapted MAPS 92.2, 115--129.)
\end{thebibliography}
\end{document}          %cgl@risc1.rug.nl
```

A.6 Template: `ttn2n3.tem`

```
\documentstyle{ttn2n3}
\begin{document}
\pagestyle{empty}
%
\begin{center}
{\Sectionfont\TeX{} and TUG NEWS}
\end{center}
\noindent \TTN\ is a newsletter for
\TeX{} and \LaTeX\ users alike\ldots
\pagestyle{myheadings}
\markboth{Vol.~0, No.~0, May 1991\qquad
 {\TTN}}{{\TTN}\qquad Vol.~0, No.~0,
 May 1991}
%
\setcounter{page}{1}
\Section{Editorial}
%
\squashedsubsection{Welcome to \TTN!}{%
   Christina Thiele \\
   Editor, Prototype \TTN \\ May 1991}

\squashedsubsection{The Truth about \TeX}%
   {Christina Thiele\\Carleton University}
%
\noindent Have you ever run across
articles or descriptions of \TeX{}\ldots

%specific items: Upcoming Events,
%               TTN index (2-column)
\end{document}
```

Appendix B: Customing

The main purpose of this section is to show the `<style>.cus` files which account for your own page size, running headers and footers.

B.1 Customing: `tugboat.cus`

```
%tugboat.cus                          Dec 93
\hoffset-.75cm\voffset-.5cm
\normalcollgt=25cm\collgt=\normalcollgt
\pagewd=18cm \twocolcolwd=8.75cm
\intercolwd=.5cm
\resetpagelgt \twocol \pageno1
\enablemetacode
\everyverbatim{\enablemetacode}
\nopunctuation\overfullrule0pt
%
\def\rtitle{\hbox to \pagewd{\small
   \issue\hfill{\it\runtit}}}
\def\rfoot{\hbox to \pagewd{\tiny
   \rlap{Draft \today}\hfill
            -\thepage-\hfill
   \llap{\copyright cgl}}}
\def\issue{MAPS 94.1}
```

```
%with
\let\small\sevenrm \let\tiny\fiverm
\let\thepage\folio
\let\ea\expandafter \let\nx\noexpand
\let\ag\aftergroup
%and macros
\def\keywords{\subhead*\sevenbf
 Keywords: *\bgroup\small\baselineskip9pt}
\def\endkeywords{\smallskip\egroup}
%
\def\abstract{\centerline{\sevenbf
 Abstract}\bgroup\quote\small
 \baselineskip9pt}
\def\endabstract{\endquote\egroup}
%
\def\quote{\endgraf\bgroup\narrower
 \smallskip\noindent}
\def\endquote{\smallskip\egroup\endgraf
 \noindent}
%
\newcount\fcnt
\def\ftn{\advance\fcnt1
 \footnote{${}^{\the\fcnt}$}}
\def\verb{}
\def\cs#1{{\tt\char92#1}}
\endinput               %cgl@risc1.rug.nl
```

B.2 Customing: `ltugboat.cus`

With respect to the page parameters I could have used `a4.sty`. Because I wanted full flexibility—not restricted to LaTeX's way—and that also the running headers and footers need adaptation I just inserted what is needed. Perhaps I should have used `\ps@cgl`? I refrained from that, because of too early generalities.

```
%ltugboat.cus                     Dec 93
\hsize8.5cm \vsize25cm
\overfullrule0pt
\textheight\vsize\textwidth\hsize
\columnsep.5cm
\advance\textwidth\hsize
\advance\textwidth\columnsep
\columnwidth\hsize
\count0=1
%
\def\rtitle{\runtit}
\def\issue{MAPS 94.1}
\let\ftn\footnote
\catcode‘\@=11
\def\@oddhead{\it\issue\hfil\rtitle}
\def\@evenhead{\it\rtitle\hfil\issue}
\def\@oddfoot{\rlap{Draft \today}\hfil
   -\thepage-\hfil\llap{\copyright cgl}}
\def\@evenfoot{\rlap{\copyright cgl}\hfil
   -\thepage-\hfil\llap{Draft \month/\year}}
\catcode‘\@=12
\endinput               %cgl@risc1.rug.nl
```

B.3 Customing: `tugproc.cus`

```
%tugproc.cus                      Dec 93
\hoffset-.75cm\overfullrule0pt
\normalcollgt24.5cm\collgt\normalcollgt
\pagewd17.5cm
\intercolwd.5cm
\resetpagelgt \twocol \pageno1
\tubpagelgt\pagelgt%Weird
\enablemetacode
\everyverbatim{\enablemetacode}
%
\def\rtitle{\hbox to \pagewd{\tenpoint
   \issue\hfill{\it\runtit}}}
%
\def\rfoot{\hbox to \pagewd{\tenpoint
   \rlap{Draft \today}\hfill-\folio-\hfill
   \llap{\copyright cgl}}}
%
\def\issue{MAPS 94.1}
%with
\let\small\sevenrm \let\tiny\fiverm
\def\quote{\endgraf\bgroup\narrower
 \smallskip\noindent}
\def\endquote{\smallskip\egroup\endgraf
 \noindent}
\newcount\fcnt
\def\ftn{\advance\fcnt1
 \footnote{${}^{\the\fcnt}$}}
\endinput               %cgl@risc1.rug.nl
```

B.4 Customing: `ltugproc.cus`

```
%ltugproc.cus                     Dec 93
\hoffset-.5cm\voffset-.25cm
\overfullrule0pt
\hsize=8.75cm \vsize=25cm
\columnsep=.5cm
\textheight\vsize \textwidth\hsize
\advance\textwidth\hsize
\advance\textwidth\columnsep
\pagewd\textwidth
\columnwidth\hsize
%
\catcode‘\@=11
\def\@oddhead{\small
   \issue\hfill{\it  \runtit}}
\let\@evenhead\@oddhead
\def\@oddfoot{{\tiny \rlap{Draft \today}
   \hfil-\thepage-\hfil
   \llap{\copyright cgl}}}
\let\@evenfoot\@oddfoot
\catcode‘\@=12
%
\setcounter{page}{1} %\count0=1
\def\issue{MAPS 93.1}
\let\ftn\footnote
\endinput               %cgl@risc1.rug.nl
```

B.5 Customing: `eurotex.cus`

```
%eurotex.cus                      Dec 93
\hoffset-.75cm \voffset-.25cm
\hsize=18cm \vsize=25cm
\textwidth\hsize \textheight\vsize
\columnwidth\hsize \linewidth\hsize
\topmargin0cm  \overfullrule0pt
\evensidemargin1cm \oddsidemargin1cm
\let\ftn\footnote
%
\catcode‘\@=11
\@colht\vsize \@colroom\vsize%Note!?!
\def\@evenfoot{\rlap{Draft \today}\rm\hfill
   -\thepage-\hfill\llap{\copyright cgl}}
   \let\@oddfoot\@evenfoot
\catcode‘\@=12
\endinput                %cgl@risc1.rug.nl
```

Appendix C: tug.ppt

The idea is to propose a preprint style which shields authors from the differences in markup in for example `tugboat.sty`, and `tugproc.sty`. It is in the same spirit as the preprint style of the AMS, and the one of Elseviers Science Publishers announced of late. It is not equivalent to `tugboat.sty`

- no options
- no spanning-columns

- various inner macros have been split-off[75]

For me it is an efficient replacement of `tugboat.sty`, provided I include my macros for
- verbatim mode suite
- bordered table
- bibliography handling, and
- some special macros required by the subject.

If this style—or at least the idea—is adopted by LUGs too, then authors can mark up their copy by this style and submit it to any LUG bulletin, without change of markup. Sounds like a nice and cooperative idea to me.

Whatever the value of `tug.ppt`, it is undoubtedly useful for educational purposes, in the sense that it shows what `tugboat.sty` is essentially all about, functionally. My case rest.

```
%Adapted from TUGboat.sty, essentially
%2-column \xcol is the column number
%within a page; ranges from 1 to \maxcols
\newcount\xcol
\newcount\maxcols
%
\newdimen\pagewd
\newdimen\colwd
\newdimen\intercolwd
%
\catcode'\@=11
% remove \outer
\def\newbox{\alloc@4\box\chardef\insc@unt}
\def\boxcs#1{\box\csname#1\endcsname}
\def\setboxcs#1{\setbox\csname#1\endcsname}
\def\newboxcs#1{\expandafter
   \newbox\csname#1\endcsname}
\newboxcs{column1}
\newboxcs{column2}
%
\def\midpage#1{\vbox{\ifnum\xcol=\maxcols
   \runhead\else\null\vskip\baselineskip\fi
   \kern2ex
   \vbox to\vsize{%
      \hbox to\pagewd{#1}\vss}
   \kern1ex
   \ifnum\xcol=\maxcols
   \runfoot\else\vskip4ex\fi}}
%
\def\runhead{\hbox to\pagewd{\sevenrm
   \issue\hfill{\it\runtit}}}
%
\def\runfoot{\hbox to\pagewd{\fiverm
   \rlap{Draft \today}\hfill--\thepage
   --\hfill\llap{\copyright cgl}}}
\def\thepage{\today}
\def\today{\ifcase\month\or Jan\or Feb\or
   March\or April\or May\or June\or July\or
   Aug\or Sept\or Oct\or Nov\or
   Dec\fi\space\number\day, \number\year}
%
\def\newcol{\endgraf\vfill\eject}
%
\def\newpage{\vfill\eject
  \loop
  \ifnum\xcol>1
    {\leavevmode\endgraf\vfill\eject}
  %\xcol is advanced in the output routine
  \repeat}
% horizontal offset of column
% from left edge of page
\newdimen\coloffset \coloffset=\z@
\def\incrcoloffset{%
  \global\advance\coloffset\colwd
  \global\advance\coloffset\intercolwd}
%
\output={\global\setboxcs{column\number\xcol}=
  \midpage{\kern\coloffset\pagebody\hfil}
  \incrcoloffset
  \ifnum\xcol=\maxcols
    \shipout\hbox{\global\xcol=\@ne
      \loop\rlap{\boxcs{column\number\xcol}}%
      \ifnum\xcol<\maxcols
          \global\advance\xcol\@ne
      \repeat
      \hbox to\pagewd{\hss}}%
    \global\advance\count0\@ne
    \global\coloffset\z@
    \global\xcol=\@ne
  \else
    \global\advance\xcol\@ne
  \fi}
\catcode'\@=12
%
\def\title*#1*{\def\thetitle{#1}}
\def\author*#1*{\def\theauthor{#1}}
\def\address*#1*{\def\theaddress{#1}}
\def\netaddress*#1*{\def\thenetaddress{#1}}
\def\article{\hrule\kern2ex\noindent
   {\bf \thetitle}\medskip\noindent
   \item{}\theauthor}
\def\endarticle{\makesignature
   \global\xcol\maxcols\vfil\eject\end}
\def\head*#1*{\goodbreak\bigskip\noindent
   {\bf #1}\medskip\noindent\ignorespaces}
\def\subhead*#1*{\medskip\noindent{\bf #1}}
\def\subsubhead*#1*{\smallskip{\bf #1}}
\def\tubissue#1(#2){\TUB~#1, no.~#2}
\def\\{\hfil\break}
\def\makesignature{\medskip
   \rightline{\hbox to.5\hsize{\strut
      \llap{$\diamond$\quad}\theauthor\hss}}
   \rightline{\vbox{\noindent
     \hsize=.5\hsize
     \theaddress\endgraf\noindent
     \thenetaddress}}}
%
% Defaults
%
\def\onecol{\maxcols=1
    \hsize=16cm
    \pagewd=\hsize
    \colwd=\hsize
    \vsize=25cm
    \maxcols=1
    \xcol=1
}
%
\def\twocol{\maxcols=2
    \hsize=8.75cm
    \colwd=\hsize
    \intercolwd=.5cm
    \pagewd=18cm
    \vsize=25cm
    \maxcols=2
    \xcol=1
}
\hoffset-1cm
\voffset-1cm
\twocol
\endinput
Contents
- OTR
```

[75] Also useful in other contexts than *TUGboat*.

```
        \boxcs...........................14
        \setboxcs........................15
        \newboxcs........................16
        \midpage..................... 21-28
        \runhead......................30-31
        \runfoot......................33-35
        \thepage.........................36
        \today........................37-40
        \newcol..........................42
        \newpage......................44-49
        \incrcoloffset................53-55
        \output.......................57-72
     - Markup structures
        \title...........................75
        \author..........................76
        \address.........................77
        \netaddress......................78
        \article......................79-81
        \endarticle...................82-83
        \head.........................84-85
        \subhead.........................86
        \subsubhead......................87
     - Miscellaneous
        \tubissue........................88
        \\...............................89
        \makesignature................90-96
     - Defaults
        \onecol.....................100-106
        \twocol.....................109-117
     - Initializations              118-120
     %
     %Split-off---inner level---are
     % - tugboat.cmn abbreviations
     % - size switching macros
     % - \quote...\endquote
     % - \bitem (essentially plain’s \item)
     % - verbatim mode macros
     % - (bibliography macros,
     %    \btable...\endbtable,
     %    ...)
     %Author: C.G van der Laan, Hunzeweg 57,
     %        9893PB Garnwerd, The Netherlands,
     %        05941-1525 cgl@risc1.rug.nl
     %History of changes
     %March 94   Essential set up
```

I have used the tug.ppt style already for my articles BLUe’s Bibliography, Transparencies and Verbatim.
A bare-to-the-bones sample is

```
%test tug.ppt       %cgl@risc1.rug.nl
\input tug.ppt
\input cgl.mac    %My inner level macros
%\input tug.fonts %Size switching macros
%\input tug.abr   %Abbreviations and logos
%
\def\runtit{tug.ppt}
\title*\runtit---{\rm the alternative}*
\author*Kees van der Laan*
\address*Hunzeweg 57, 9893PB\\
         Garnwerd,
         The Netherlands*
\netaddress*cgl@risc1.rug.nl*
\def\issue{MAPS 94.1}
\article
\keywords ppt, education.\endkeywords
\abstract A preprint style for TUG publications
\endabstract
%
\head*Why?*
Although it seems misplaced at first
sight---apart from tugboat,sty and
tugproc.sty---it can be worthwhile for
the \TeX{} community at large to provide
for a preprint style, such that
each author can submit a paper
in preprint style, while the editorial
board at hand substitutes the preprint
style by the style needed for the
concrete bulletin. This approach is
similar to AMS’ approach.
\endarticle
```

Appendix D: Contents

The structure of this article is a bit complex because of the modular treatment of the various style files. The article begins with a general part followed by specific parts, each devoted to a style file. At the end there is again a general part which contains ‘Looking back’ and the like. The appendices reflect the same splitting in parts, where each part is again devoted to a specific style file.