LaTeX class writing for wizard apprentices

Boris Veytsman

Abstract

A number of excellent articles explain LaTeX class writing for beginners (Hefferon, 2005; Flynn, 2007; Pakin, 2008; Mansfield, 2008). Of course, true wizards do not need any instructions. This paper is intended for those TeXnicians who are no longer beginners, but may not (yet) qualify as wizards. It discusses some tips and tricks of the trade: packages you may want to use and packages you must be compatible with; why the first two pages of a book take 60% of your time; what is wrong with the LaTeX sectioning interface but why you better stick with it, and more. It is based on the experience of the author in writing LaTeX classes for various customers (Veytsman, 2008c).

1 Audience and scope

The New Hacker's Dictionary (Raymond, 1996) defines the word "user" in the following way:

> **user:** *n.*
>
> 1. Someone doing "real work" with the computer, using it as a means rather than an end. Someone who pays to use a computer. See *real user.*
> 2. A programmer who will believe anything you tell him. One who asks silly questions. [GLS observes: This is slightly unfair. It is true that users ask questions (of necessity). Sometimes they are thoughtful or deep. Very often they are annoying or downright stupid, apparently because the user failed to think for two seconds or look in the documentation before bothering the maintainer.] See *luser.*
> 3. Someone who uses a program from the outside, however skillfully, without getting into the internals of the program. One who reports bugs instead of just going ahead and fixing them.
>
> The general theory behind this term is that there are two classes of people who work with a program: there are implementors (hackers) and *lusers.* The users are looked down on by hackers to some extent because they don't understand the full ramifications of the system in all its glory. (The few users who do are known as real winners.) The term is a relative one: a skilled hacker may be a user with respect to some program he himself does not hack. A LISP hacker might be one who maintains LISP or one who uses LISP (but with the skill of a hacker). A LISP user is one who uses LISP, whether skillfully or not. Thus there is some overlap between the two terms; the subtle distinctions must be resolved by context.

In our TeX world the distinction between users and implementors is sometimes represented as a distinction between users and TeXnicians. The latter word was introduced by DEK himself (Knuth, 1994). A TeXnician is a person who helps other people to use TeX. This paper is intended for them.

More specifically, here we discuss writing LaTeX classes. I hope some of these issues might be of interest for the macro writers working with other formats, but I personally have been dealing mostly with LaTeX.

The TeXnicians working with macro packages can be divided into three groups. First, there are true wizards (see the definition and discussion in *New Hacker's Dictionary*). They can easily and confidently write pages of TeX code with all manner of `\expandafter` and `\futurelet` uses. Clearly these people do not need to be taught how to write classes (at least by me!).

Next, there are beginners. I do not mean here beginning *users,* just the opposite: this group largely consists of experienced users trying themselves in class writing. There is good and useful literature for this category of TeXnicians; I heartily recommend the papers by Hefferon, 2005; Flynn, 2007; Pakin, 2008; Mansfield, 2008.

The third group consists of people who already authored one or several classes, know how to use the DTX format, have a dog-eared copy of the *Companion* (Mittelbach, Goossens, Braams, Carlisle, and Rowley, 2004) and can recite paragraphs from the *Class Guide* (2006). For the lack of better term we will call such people *wizard apprentices.* I am proud to belong to this group myself. One of my longstanding complaints has been the relative dearth of literature intended for this category of macro writers. The great book by Eijkhout, 2007, is one of the rare exemptions from this rule. This article is also intended to partially fill this need.

I provide here some anecdotes and snippets from my experience as a LaTeX macro writer. If it seems rather subjective and opinionated, you, the reader, are welcome to offer your own point of view. I am certainly open to suggestions and critique from my fellow apprentices, as well as from the wizards and the beginners.

2 LaTeX interfaces

It is well known that the design of the standard LaTeX classes is not optimal from the typographer's point of view. The fact that its user interfaces also leaves much to be desired is probably less appreciated.

Let us consider, for example, the LaTeX sectioning commands. The standard `\chapter` command and its sisters `\section`, `\subsection`, etc. have two arguments: the mandatory title and the optional short title. The latter, if present, is used for the table of contents and headers. This means that the entry in the table of contents must be the same as the headers. This is not what we want in most cases. It is more common for modern books to use the "long" title in the table of contents and the "short" one in the headers.

The starred variants `\chapter*`, `\section*`, etc. have their own problems. First, the star is overloaded: it signifies *both* that this heading is not numbered *and* that it does not produce an entry in the table of contents. This is a wrong idea in most cases. Such unnumbered subdivisions as *Bibliography* and *Index* should be mentioned in the table of contents. The same is true for front matter headings like *Foreword,* which are usually not numbered, but nevertheless belong in the table of contents. One of the frequently asked questions in the Usenet newsgroup `comp.text.tex` is "how to add the bibliography to the TOC?" Of course there are packages which achieve this, but they are correcting a design misfeature which should not be there in the first place. Another problem is the fact that the starred sectioning commands do not have the optional argument (and do not change the headers in the standard classes). Again, in most cases we do want to change the headers for unnumbered subdivisions and thus need a way to set up a short version of the title.

It is reasonable for a class writer to correct these misfeatures and create a more rational interface design. For example, most of the problems discussed above can be easily solved in a new class. However, one must be very careful when changing the interfaces — even if it is tempting to do so. First, the users are accustomed to the LaTeX design, and drastic changes might be too difficult for them. Second, many computer editors are "LaTeX-aware", and offer users ready-made templates. Third, some useful packages like *hyperref* (Rahtz and Oberdiek, 2006) redefine LaTeX commands, and expect certain interfaces to be there.

The following example from my experience illustrates this point. Books published by *Nostarch Press* have so-called *circular art*: a small round picture in the beginning of each chapter. In the preliminary version of the class for this publisher I used the following interface with one optional and two mandatory arguments:

```
\chapter[⟨Short Title⟩]{⟨Long Title⟩}{⟨Artwork⟩}
```

This was a disaster. It turned out to be difficult for me to remember that `\chapter` now has *two* mandatory arguments, so I got a slew of errors during testing. Moreover, *hyperref* refused to understand this syntax, and I decided that patching it would be too cumbersome. In the end I opted for another design (Veytsman, 2008b):

```
\chapter[⟨Short Title⟩]{⟨Long Title⟩}
\chapterart{⟨Artwork⟩}
```

3 Compatibility issues

In the previous section we briefly touched on the issue of compatibility. Let us discuss it more thoroughly.

The LaTeX world can be described as a motley collection of packages written by different authors with various philosophies, goals, design ideas and skill levels. Nobody ever guaranteed that these packages would work together. The fact that TeX has no concept of namespace makes the combination of these packages even more daunting.

To tell the truth, though, the situation is better than it could be. Most packages are compatible, and the authors usually take care to patch them if they turn out not to be. Still, sometimes interactions between packages lead to unexpected results.

Some publishers, when accepting LaTeX manuscripts, restrict the authors' choice of packages. Nevertheless there are always situations when an author or an editor really needs a package.

A macro writer should expect the users to load some packages. A good policy is to proactively test the class with the most popular ones and mention such "approved" packages in the documentation. In some cases it makes sense to automatically load these packages from the class.

If an electronic publication is intended (which is almost always the case nowadays), *hyperref* (Rahtz and Oberdiek, 2006) is usually called. This package has many settings changing the appearance of links (the default ones are almost never satisfactory). However, setting them presents the following problem. This package usually should be loaded last, after others. If you load it from your class, the packages called by the users may not work properly.

There are two ways to solve this problem. First, you can write a wrapper package that calls *hyperref*, sets it up and even patches it if necessary. I adopted this approach for the *nostarch* class (Veytsman, 2008b). Second, you can automatically check at the beginning of the document whether *hyperref* is loaded, and set it up if necessary with a construction like this:

```
\AtBeginDocument{%
  \@ifpackageloaded{hyperref}{%
    \hypersetup{⟨Your customization⟩}}{}}
```

I used this method for the class for an online journal *Philosophers' Imprint* (Veytsman, 2007).

Most people use the graphics bundle (Carlisle, 2005) and sometimes other graphics solutions for illustrations. If your users are mathematicians, they probably are going to load *amsmath* (2002). Programmers usually call *listings* (Heinz and Moses, 2007). Many people working in "natural sciences" need *natbib* (Daly, 2009). You may want to test your class with these packages. In general, a class writer should understand the users' ways of doing things and anticipate their actions.

It is difficult enough to ensure compatibility with the plethora of LaTeX packages, but in fact the problem is even more complex, because these package are not frozen. Sometimes when a user upgrades a package, it breaks many existing documents. This requires constant vigilance from the class author.

4 Some useful tools

Besides creating problems for a class author, the great variety of LaTeX packages has positive value too. Some packages turn out to be great tools for macro writers. In this section I list some tools I often use.

The package *ifpdf* (Oberdiek, 2006) provides the information whether the document is compiled via the DVI route or the direct PDF output is chosen. This is useful, for example, if the class uses a non-standard paper size (this is common for books). We can tell LaTeX the paper size, but to communicate it to e.g. *dvips*, we must instruct the user to put the corresponding option in the program call. If the direct PDF route is chosen, however, we can make the user's life slightly simpler with the commands like these (taken from the *memoir* code (Wilson, 2004)):

```
\ifpdf\relax
  \pdfpageheight=\paperheight
  \pdfpagewidth=\paperwidth
  \pdfvorigin=1in
  \pdfhorigin=1in
\fi
```

The package *geometry* (Umeki, 2008) is quite handy for easy setup of paper size and margins.

The package *fancyhdr* (van Oostrum, 2004) supports defining complex headers and footers, with or without decorative rules.

The package *caption* (Sommerfeldt, 2007) is useful for setting up captions for tables, figures and other floats.

Nowadays many designers prefer ragged layout of the copy. The package *ragged2e* (Schröder, 2003) helps to implement it in a rational way.

In many cases the "house style" includes specific rules for the bibliography. Then the *custom-bib* package (Daly, 2003) can help to create a customized BibTeX style according to the requirements.

There are many more tools that should be in a class writer's toolbox. A constant monitoring of Usenet groups and CTAN announce lists helps to keep abreast of the TeX development. However, there is always something to learn. When presenting this talk at TUG 2009, I complained that the LaTeX `\@addtoreset` command does not have a counterpart that removes some counters from the reset list. Barbara Beeton immediately recalled a small package (Carlisle, 1997) that provides a very useful command `\@removefromreset`.

Another matter related to the reuse of somebody else's code is whether to start a new class from scratch or to load a base class and then to redefine it as necessary. I guess this is a matter of taste. I myself prefer the second approach, but often it turns out in the end that the new class redefines so many macros that there is not much left from the original class.

5 These first pages ...

The cover, title and copyright pages are only a small fraction of the copy. Nevertheless they require a large percentage of a class author's effort. My experience shows that macros for these pages take about 60% of the total time for articles and 80% of the total time for books.

These pages require highly formal typesetting presenting structured information. This is easy to note in books (see the title and copyright pages for the *nostarch* class (Veytsman, 2008b)), but it is also true in reports: see the samples for the *erdc* class written for the reports of US Army Corps of Engineers (Veytsman, 2009).

For these macros we can dispense with the advice of Section 2 and change the standard LaTeX interfaces, since they are woefully inadequate, and most classes completely redesign front matter macros anyway. However there is one idea of front matter macros in standard LaTeX which is worth keeping. It is the idea of the separation of macros for *collecting data,* like `\author`, `\title`, etc. — and the macros for typesetting like `\maketitle`. This approach allows entering the front matter data in any order, and to start typesetting only when we know which data are present and how large they are.

The commands for collecting data store them in internal macros. There are three kinds of such commands, which we discuss below.

The commands of the first kind are very simple. They have just one argument, and they store it in an internal macro. For example, the date in standard LaTeX can be defined as:

```
\def\date#1{\gdef\@date{#1}}
\date{\today}
```

In a more complex case the command has two arguments: the mandatory one and the optional one, with the usual convention that in the absence of the optional argument the mandatory one is used instead. As an example let us consider the `\title` command. In standard LaTeX it has just one argument, but we might want to have a full title for the title page and a short one for running heads, etc. This is a common enough case for books (Veytsman, 2008b), but can be found in articles if the journal uses article titles in headers (Veytsman, 2007; Veytsman, 2008a). With this command we can say something like

```
  \title[Robinson Crusoe]{%
    The Life and strange Surprizing Adventures
    of Robinson Crusoe of York, Mariner:
    Who lived Eight and Twenty Years, all
    alone in an un-inhabited Island on the
    coast of America, near the Mouth of the
    Great River of Oroonoque; Having been cast
    on Shore by Shipwreck, where-in all the Men
    perished but himself. With An Account how
    he was at last as strangely deliver'd by
    Pyrates. Written by Himself}
```

(by the way, the long title is the actual title of Defoe's book!). This effect can be achieved by the `\@ifnextchar[` macro:

```
\def\title{\@ifnextchar[{%
    \title@i}{\title@ii}}
\def\title@i[#1]#2{%
  \gdef\@shorttitle{#1}\gdef\@title{#2}}
\def\title@ii#1{%
  \title@i[#1]{#1}}
```

The result is that the internal macro `\@title` gets the full title, while the macro `\@shorttitle` gets the short title.

The most complex case is *cumulative macros:* each command can be repeated several times, and the consecutive macros add to the stored information. Suppose for example that we have several groups of authors with shared affiliations. Than a natural syntax (following the ideas of American Mathematical Society classes, Downes and Beeton, 2004) to enter the information is the following:

```
\author{A.U.~Thor \and C.O.R.~Respondent
  \and C.O.~Author}
\affiliation{Construction Engineering
  Research Laboratory\\
  U.S. Army Engineer Research and
  Development Center\\
  2902 Newmark Drive\\
  Champaign, IL 61826-9005}
\author{John~M.~Smith}
\affiliation{Coastal and Hydraulics
  Laboratory\\
  U.S. Army Engineer Research and
  Development Center\\
  3909 Halls Ferry Road\\
  Vicksburg, MS 39180-6199}
```

This can be achieved in the following way:

```
\def\author#1{%
  \ifx\@empty\@authors
     \gdef\@authors{#1}%
  \else
    \g@addto@macro{\@authors}{\and#1}%
  \fi
  \ifx\@empty\@addresses
     \gdef\@addresses{\author{#1}}%
  \else
     \g@addto@macro{\@addresses}{%
       \par\author{#1}}%
  \fi}

\def\affiliation#1{%
  \ifx\@empty\@addresses
     \gdef\@addresses{#1\par}%
   \else
      \g@addto@macro{\@addresses}{%
        #1\par\vspace{\baselineskip}}%
   \fi}
```

As the result of these commands we have two internal macros: `\@authors` keeping the authors separated by `\and`, and `\@addresses` keeping the authors and their affiliations. Now we need to substitute `\and` inside them by the proper punctuation. Here the command `\andify` from Downes and Beeton, 2004, is very handy. By default it uses American punctuation (Tom, Dick, and Harry), but it has options for other variants. I used this possibility in packages *ijmart* and *erdc* (Veytsman, 2008a; Veytsman, 2009). The latter package provided an interesting challenge: it required the full list of authors without affiliations separated by commas and *and* on the cover, and separate lists of authors with shared affiliations on the title page. Having a separate macro with just the authors' names helped to do this properly.

Another interesting thing to do is to automatically set up metadata for PDF output. Many publishers do not specify this in their requirements, but a class writer should know better. In any case this must be done when online publication is intended. As discussed above, we can determine whether *hyperref* is loaded and whether the PDF output is specified. If these conditions are true, then the construction of the title page should include lines like these:

```
\hypersetup{pdfauthor=\@author,%
  pdftitle=\@title,%
  pdfsubject=\@subject,%
  pdfkeywords=\@keywords}
```

6 Final remarks and conclusion

In the previous sections we discussed the computer-related aspects of class writing. There are, however, even more important human aspects.

It is essential to have a good interaction with the typographic designer and the users. A good typographer can explain what is required from the class and convey her or his ideas to the class writer. Some class writers try to double as designers; I always felt that the design requires years of training and apprenticeship. Just the fact that you can code complex things in TEX does not mean these things are beautiful or proper.

Good users can test the code and write sensible bug reports. This is also essential for the success of the class.

If you have good typographic artists and patient users, class writing is fun and rewarding. I personally enjoyed writing the macros for my LATEX classes.

Acknowledgements

The author is grateful to the users and typographic designers of his LATEX classes for their patience and tireless work, to the audience of TUG 2009 for many interesting suggestions.

References

User's Guide for the amsmath Package (Version 2.0). American Mathematical Society, 2002. http://mirror.ctan.org/macros/latex/required/amslatex/math/amsldoc.pdf.

LATEX 2ε for Class and Package Writers. LATEX3 Project, 2006. http://mirror.ctan.org/macros/latex/doc/clsguide.pdf.

Carlisle, David. *remreset Package*, 1997. http://mirror.ctan.org/macros/latex/contrib/carlisle.

Carlisle, D. P. *Packages in the 'Graphics' Bundle*, 2005. http://mirror.ctan.org/macros/latex/required/graphics.

Daly, Patrick W. *Customizing Bibliographic Style Files*, 2003. http://mirror.ctan.org/macros/latex/contrib/custom-bib.

Daly, Patrick W. *Natural Sciences Citations and References (Author-Year and Numerical Schemes)*, 2009. http://mirror.ctan.org/macros/latex/contrib/natbib.

Downes, Michael, and B. Beeton. *The amsart, amsproc, and amsbook document classes.* American Mathematical Society, 2004. http://mirror.ctan.org/macros/latex/required/amslatex/classes.

Eijkhout, Victor. *TEX by Topic.* Lulu, 2007. http://eijkhout.net/texbytopic/texbytopic.html.

Flynn, Peter. "Rolling Your Own Document Class: Using LATEX to Keep Away From the Dark Side". *TUGboat* **28**(1), 110–123, 2007. http://tug.org/TUGboat/Articles/tb28-1/tb88flynn.pdf.

Hefferon, Jim. "Minutes in Less Than Hours: Using LATEX Resources". *TUGboat* **26**(3), 188–192, 2005. http://tug.org/TUGboat/Articles/tb26-3/tb84heff.pdf.

Heinz, Carsten, and B. Moses. *The Listings Package*, 2007. http://mirror.ctan.org/macros/latex/contrib/listings.

Knuth, Donald Ervin. *The TEXbook.* Computers & Typesetting A. Addison-Wesley Publishing Company, Reading, Mass., 1994. Illustrations by Duane Bibby.

Mansfield, Niall. "How to Develop Your Own Document Class — Our Experience". *TUGboat* **29**(3), 356–361, 2008. http://tug.org/TUGboat/Articles/tb29-3/tb93mansfield.pdf.

Mittelbach, Frank, M. Goossens, J. Braams, D. Carlisle, and C. Rowley. *The LATEX Companion, Second Edition.* Addison-Wesley Series on Tools and Techniques for Computer Typesetting. Addison-Wesley Professional, Boston, 2004.

Oberdiek, Heiko. *The ifpdf Package*, 2006. http://mirror.ctan.org/macros/latex/contrib/oberdiek.

Pakin, Scott. "Good Things Come in Little Packages: An Introduction to Writing .ins and .dtx Files". *TUGboat* **29**(2), 305–314, 2008. http://tug.org/TUGboat/Articles/tb29-2/tb92pakin.pdf.

Rahtz, Sebastian, and H. Oberdiek. *Hypertext Marks in LaTeX: a Manual for Hyperref*, 2006. `http://mirror.ctan.org/macros/latex/contrib/hyperref`.

Raymond, Eric S., editor. *New Hacker's Dictionary.* The MIT Press, Boston, MA, third edition, 1996. `http://catb.org/jargon`.

Schröder, Martin. *The ragged2e Package*, 2003. `http://mirror.ctan.org/macros/latex/contrib/ms`.

Sommerfeldt, Axel. *Typesetting Captions with the caption Package*, 2007. `http://mirror.ctan.org/macros/latex/contrib/caption`.

Umeki, Hideo. *The geometry Package*, 2008. `http://mirror.ctan.org/macros/latex/contrib/geometry`.

van Oostrum, Piet. *Page Layout in LaTeX*, 2004. `http://mirror.ctan.org/macros/latex/contrib/fancyhdr`.

Veytsman, Boris. *Typesetting Articles for the Online Journal* Philosophers' Imprint, 2007. `http://mirror.ctan.org/macros/latex/contrib/philosophersimprint`.

Veytsman, Boris. *LaTeX Class for The Israel Journal of Mathematics*, 2008a. `http://mirror.ctan.org/macros/latex/contrib/ijmart`.

Veytsman, Boris. *LaTeX Style for* No Starch Press, 2008b. `http://mirror.ctan.org/macros/latex/contrib/nostarch`.

Veytsman, Boris. "Observations of a TeXnician for Hire". *TUGboat* **29**(3), 484, 2008c. `http://tug.org/TUGboat/Articles/tb29-3/tb93abstracts.pdf`.

Veytsman, Boris. *LaTeX Style for Technical Information Reports of the Engineer Research and Development Center, US Army Corps of Engineers*, 2009. `http://mirror.ctan.org/macros/latex/contrib/usace`.

Wilson, Peter. *The Memoir Class for Configurable Typesetting*, 2004. `http://mirror.ctan.org/macros/latex/contrib/memoir`.

◇ Boris Veytsman
Computational Materials Science Center, MS 6A2
George Mason University
Fairfax, VA 22030
`borisv (at) lk dot net`