The PracTEX Journal, 2006, No. 2
Article revision 2006/05/16

Travels in TEX Land: LATEX for Productivity in Book Writing

David Walden

Abstract In this column in each issue I muse on my wanderings around the TEX world. My column in this issue summarizes why I use LATEX and gives examples of some productivity benefits of using LATEX to write books.

1 Why I use LATEX

Two reasons typically given for using LATEX are for its math support and for very nice looking typesetting. Neither of these is particularly important to me: I rarely have any math in my writing (but it is nice to be able to handle it easily in those rare cases where I do have it); I have a pretty undiscerning eye when it comes to typesetting, and what LATEX produces is more than good enough for me.

The things that matter most to me about LATEX are:[1]

1. its programmability and modularity
2. that I get to use a powerful editor with it
3. that the mark-up is clearly visible to me and can be changed directly with a text editor
4. its capabilities for explicitly noting cross-references, maintaining bibliographies, and automatically numbering chapters, sections, figures, tables, footnotes, etc., which permit easy reorganization of text within documents and reuse in other documents
5. its relatively slow pace of change and great concern among the developers for backwards compatibility

In other words, my use of LATEX is primarily about productivity. (Of course, there are certain limitations on this productivity such as when I finish writing a book using LATEX and the publisher tells me I must convert the text to Word and the figures to PowerPoint slides for input into the compositor's typesetting system.)

1. I'll discuss these more in a future paper; this paper will primarily concentrate on productivity methods I use related more or less specifically to book writing.

2 Writing books using LaTeX

Much of my work using LaTeX is on book length documents. For these I have compiled a more or less standard set of techniques that I feel help me be more efficient. I don't claim that the techniques I use are the techniques of a master; in fact, I view myself as an intermediate user of LaTeX — I know enough to make LaTeX jump through a few simple hoops, but not enough to know if my approaches are recommended or if they include some bad habits.

In my experience, publishers don't think much about the design of a book until they have the completed manuscript in hand. Since I use LaTeX to develop the original manuscript, I have to make lots of temporary design decisions, and I want to be able to change these decisions with a minimum of work when the publisher does begin to deal with the design. Also, I am currently working on a book that I will be self publishing, and settling on the design for this book is an iterative, experimental process where it is even more important to be able to make changes throughout the book (for instance, to the style of figure captions) with minimal work. My experience, however, should not prevent you from checking if the publisher of your document already has a standard style and perhaps even a LaTeX class file that you can use from the the outset of your writing. In any case, my emphasis here is *not* on the methods of representing preferences for appearance; my emphasis is on methods for easily and repeatedly *changing* the overall document appearance as well as on other methods for working efficiently on large documents.

Some of what I am about to describe for working efficiently on books or other long documents is probably already well known to many readers; perhaps you can make suggestions for how I might do things better.

(At several points in the following, I have included in parentheses discussions of basic TeX and LaTeX issues that reviewers and pre-publication readers have asked me about that are not actually on the subject of book-writing productivity. Perhaps these parenthetical notes should have been footnotes, but I was too lazy to deal with the need for alternatives to `\verb` in footnotes.)

2.1 Include files

Suppose I am working on a book entitled *Breakthrough Management*, as I have been recently. I created a top level file named `bt.tex` with the following contents:

```
\documentstyle{btbook}
```

```
\begin{document}
\include{titlepages}
\include{preface}
\include{surviving}   % a chapter
\include{rapid}       % another chapter
. . .                 % more chapters
\include{acknowledgements}
\include{bibliography}
\include{bio}
\include{index}
\end{document}
```

The text from included files appears to LaTeX as if it was in the file `bt.tex` in place of the `\include` commands. In this way, I contain the text related to each chapter and other parts of the book in its own file. I let LaTeX take care of numbering the chapters and figures (or whatever) within chapters. If I later decided to change the order of chapters, I just change the order of the `\include` commands in the `bt.tex` files, and LaTeX automatically renumbers everything.

To work on one chapter at a time, my file `bt.tex` evolved to include many `\includeonly` commands, e.g.,

```
\documentstyle{btbook}
%\includeonly{preface}
%\includeonly{surviving}
\includeonly{rapid}
%includeonly{surviving,rapid}
. . .
\begin{document}
\include{titlepages}
\include{preface}
\include{surviving}   % a chapter
\include{rapid}       % another chapter
. . .                 % more chapters
\include{acknowledgements}
\include{bibliography}
\include{bio}
\include{index}
\end{document}
```

In the above example, only the file `rapid.tex` gets compiled when I run LaTeX on the file `bt.tex`. In this 10 chapter book I had a couple of dozen `\includeonly` commands in the bt.tex file that I could comment in and out to work on each chapter individually and with various combinations of related chapters.

(Because the `\include` commands result in text being typeset, they must go after the `\begin{document}` command. The `\includeonly` commands must go in the preamble or else LaTeX complains.)

2.2 Custom class file

I have created a file `btbook.cls` which is my own personal class file for this particular book. This file is processed when LaTeX sees the `\documentstyle{btbook}` command at the beginning of the file `bt.tex`. The first three lines of the file

```
\NeedsTeXFormat{LaTeX2e}[1994/12/01]
\ProvidesClass{btbook}[2006/01/21 Breakthrough management book class]
\LoadClass{book}
```

define the class for this book to be named btbook and to be an augmentation of the LaTeX book class.

The rest of the lines of the file are read and executed when LaTeX is run as if they were lines of text immediately following the `\documentclass` command in the `bt.tex` file.

(If your publisher already provides a LaTeX class file, you can still collect all of the sorts of things I describe below in their own file and `\input` that file in the preamble rather than just putting all these things directly in your preamble. I prefer not to have much in my preamble beyond the `\includeonly{...}` commands that I am constantly commenting in and out.)

2.3 Packages

Next in the class file come the list of packages I use for writing this book.

```
\RequirePackage{mathpazo}  % Palatino is basic roman font
\RequirePackage[scaled=.95]{helvet}  % Helvetica is sans serif font
\RequirePackage{courier}   % Courier is typewriter font
\RequirePackage{graphicx}  % for including images
\RequirePackage{url}       % for formatting URLs
\RequirePackage[figuresright]{rotating} %to be able to rotate figures
```

```
\RequirePackage{lettrine}  % for dropped caps
\RequirePackage{paralist}  % for tighter list spacing
       \setlength{\pltopsep}{.05in}
\RequirePackage{comment}   % for comment environment
\RequirePackage{dw-endnotes} % for endnotes with reformatted numbers
\RequirePackage{setspace}  %\doublespacing
```

When I find I need to use another package, I add another `\RequirePackage` line to this list. (As I understand it, `RequirePackage` does the same job as `usepackage` except it doesn't allow the same package to be loaded twice which apparently might cause problems in some cases.)

Notice that the package name in one case includes the characters `dw-`. This is my convention for noting a package that I have modified. In such cases, the file of the modified package is in the same directory with the rest of the files for this book or in the local changes part of my `texmf` data structure. I seldom understand a package I am modifying; I typically use a hit and miss approach to chang stuff until I get the results I want.

Copy editors who edit on hard copy like double spacing, and I can provide that with a one character change — uncommenting the `\doublespacing` command on the last line above that loads the setspace package.[2]

2.4 Miscellaneous useful macros

The following macros provide a few capabilities I use relatively frequently.

```
\newcommand{\Dash}{\thinspace---\thinspace} % space around em-dashes
\newcommand{\CK}[1]{\textbf{CK #1}}  % mark text that needs checking
\newcommand{\manote}[1]{% marginal note to myself
\marginpar{\scriptsize To do:\\
#1}}
\newcommand{\partitle}[1]{\medskip\noindent\textbf{#1}} % cut-in title

\let \Originalurl = \url  %change function of \url command
\renewcommand{\url}[1]{{\fontsize{10.7pt}{13.05pt}\Originalurl{#1}}}
```

For some documents I have worked on, I have had many more such miscellaneous useful macros.

2. The answer to the second question of the Ask Nelly column in this issue describes a different way to cause double spacing.

Anyone trying to improve productivity using LaTeX who doesn't already define his or her own macros should learn to do so. User-defined macros allow significant improvements in efficiency. For instance, the first macro above defines the command `\Dash{}` to be an abbreviation for the string of characters `\thinspace---\thinspace` which results in an em-dash being typeset with a *little* bit of space on each side of it, as in aaa — bbb. It is less characters and probably more reliable to type `\Dash{}` many times in a book than it is to type `\thinspace---\thinspace{}` many times. In my view, however, the greater benefit of defining the `\Dash` command comes when my publisher tells me that its style is closed-form em-dashes (no space on each side, i.e., aaa—bbb) or a more open form (aaa — bbb). To implement either of these changes *throughout* the book, I merely redefine `\Dash`, e.g.,

```
\newcommand{\Dash}{---} % no spaces around em-dashes
```

or

```
\newcommand{\Dash}{ --- } % full spaces around em-dashes
```

and recompile my document. Containing such style conventions within a few lines of a large document and being able to change the style throughout the document with only a few key strokes is an enormous advantage. (I'll give a more complex example of such containment when I discuss macros for figures and tables below.)

To redefine a command that already exists in LaTeX or has been defined by a package that has already been loaded, for instance to define a variation on `\url` as I do in the last two lines of my group of miscellaneous useful macros, I have to use the `\renewcommand` command. The `\renewcommand` works just like `\newcommand` except that LaTeX doesn't complain with `\renewcommand` if I try to give a definition to a command that already exists — a good thing to be warned about when one uses `\newcommand`. (In the next subsection I give another redefinition example — redefining `\footnote`).

2.5 Footnotes and endnotes

In the case of `\RequirePackage{dw-endnotes}`, I am using the `endnotes` package, modified slightly to change the format of the note numbers.

Typically, I put footnotes on the bottom of text pages where they are referenced, at least while I am drafting chapters and want to be able to see the notes without having to turn a bunch of pages. However, publishers tend not to like having footnotes — it makes a book look too academic to be popular, in their view. Thus, before actually publication, I often find myself converting all my footnotes to endnotes. The next commands in my class file do this.

```
\renewcommand{\footnote}{\endnote} %comment out to not have end notes

\newcommand{\dumpendnotes}
{\medskip
\begingroup
\setlength{\parindent}{0pt}\setlength{\parskip}{1ex}
\renewcommand{\enotesize}{\normalsize}
\theendnotes\endgroup \setcounter{endnote}{0}}
```

First, the `\footnote` command is redefined to be the `\endnote` command; this avoids me having to replace every instance of `\footnote` with `\endnote`. Then the class file defines a command (`\dumpendnotes`) that can go at the end of each chapter to dump the chapter's endnotes, formatted as I want them to me. If `\dumpendnotes` was already defined in LaTeX or some other package, LaTeX would warn me because I didn't do the definition with `\renewcommand`.

2.6 Formatting figures and tables

The next set of commands in the class file have to do with changing the format of figure and table captions without actually modifying a LaTeX or package file. The LaTeX default does not use bold face for captions and uses a period rather than a hyphen between the the chapter number and figure number within a chapter. The following changes patch LaTeX to follow my preference for bold face and hyphens.

```
\long\def\@makecaption#1#2{%
  \vskip\abovecaptionskip
  \sbox\@tempboxa{\textbf{#1}. \textbf{#2}}%
  \ifdim \wd\@tempboxa >\hsize
    {\textbf{#1}. \textbf{#2}\par}
  \else
    \global \@minipagefalse
    \hb@xt@\hsize{\hfil\box\@tempboxa\hfil}%
  \fi
  \vskip\belowcaptionskip}
\renewcommand \thefigure
```

```
     {\ifnum \c@chapter>\z@ \mbox{\thechapter-\fi\@arabic\c@figure}}
\renewcommand \thetable
     {\ifnum \c@chapter>\z@ \mbox{\thechapter-\fi\@arabic\c@table}}
%end of commands to change style of figures and table captions
```

(I do not have to bracket these lines, top and bottom, with `\makeatletter` and `makeatother` commands as I would have to if this patch was in the preamble of my document; the at-sign is a letter by default in class files and packages. Some readers may be back a step, at the question of, "What is the about at-signs anyway?": The answer is that the files for basic LaTeX, for class files, and for other packages are full of macros names that include an at-sign (@), e.g., a macro named `\@makecaption` is defined at the beginning of the above example. My understanding is that an at-sign is used in low level programming of LaTeX, class files, and packages to create macro names that can't accidentally conflict with names that may be defined by users not doing such LaTeX "systems programming." An at-sign is normally not a letter and thus cannot be part of a macro name. However, in the above example I want to patch low level LaTeX code that includes at-signs in its macro names; if I was trying to make this patch in my preamble (as I used to do before I learned to make some patches in a personal class file), I would have to tell LaTeX to temporarily turn at-signs into letters, make the patch, and then turn them back into non letters (other) so the rest of my program could use @ in the normal way where it is not a special character of any kind.)

Perhaps there is a caption package that would allow such changes without patching LaTeX, but I was shown how to make this patch a few years ago and it works, so why bother trying to find and learn a new package?

Next in my class file come a set of definitions for commands I use to include graphics. I seldom insert `\begin{figure}` and `\end{figure}` commands directly into my documents and insert `\begin{table}` and `\end{table}` commands only a little more often. It is inevitable that, before I am done with a big document, I will want to change the formatting relating all figure and tables — perhaps several times. Thus, I use macros for inserting almost all figures (or tables) such that I can make changes to formatting relating to the figures by making changes to only a few lines in the relevant macros.

```
\newcommand{\figfiletype}{pdf}
\graphicspath{{figures/}} %tell LaTeX a directory path for where to find figures

\newcommand{\snfig}[3]{   %scaled numbered figure
   \begin{figure}[htbp]  %drop htb for single page figures and uncomment following
%\vbox to \vsize{%
```

```
    \hfil\scalebox{#3}{\includegraphics{#2.\figfiletype}}\hfil
   \caption{\label{fig:#2}#1 \texttt{\small[#2 #3]}}
%\vfil
%}
   \end{figure}
   }

\newcommand{\sntab}[3]{   %scaled numbered tables
   \begin{table}[thbp]
%\vbox to \vsize{%
   \centering
   \caption{\label{tab:#2}#1 \texttt{\small[#2 #3]}} \smallskip
    \scalebox{#3}{\includegraphics{#2.\figfiletype}}
    \label{tab:#2}
%\vfil
%
   \end{table}
   }

\newcommand{\unfig}[2]{   %scaled unnumbered figure
   \begin{figure}[htbp]
    \hfil\scalebox{#2}{\includegraphics{#1.\figfiletype}}\hfil
   \label{fig:#1}\centerline{\texttt{\small[#1 #2]}}
   \end{figure}
   }

\newcommand{\swsnfig}[3]{   %sideways scaled numbered New figure
   \begin{sidewaysfigure}
    \centering
    \scalebox{#3}{\includegraphics{#2.\figfiletype}}
   \caption{\label{fig:#2}#1 \texttt{\small[#2 #3]}}
   \end{sidewaysfigure}
   }
```

For instance, the macro `\snfig` above takes three arguments. The text for a figure caption, the unique part of the file name for the graphic to be included, and a scale factor for the graphic, e.g.,

```
   \snfig{This is the caption}{figure3-31}{.83}
```

The full name of the file to be included is the concatenation of the part of the file name that came from the second argument of the macro call, the directory that is specified by the `\graphicspath` command (an option of the `\graphicx` package) as the place LaTeX searches for figures, and the `\figfiletype` definition as the file name extension. The latter is useful because sometimes all of my figures are `.eps` files and sometimes they are `.pdf` files, and sometimes I switch between these two formats at different times in the production of the book. (When using `.eps` format, I compile using LaTeX and a dvi-to-pdf conversion; when using `.pdf` format, I use pdfTeX to compile. If the graphic format was changing from file to file within the document, I would instead specify the format as another argument to the `\snfig` command.)

While I am drafting and revising a book manuscript, I want to be able to look at a figure in the printed output and know what file I need to modify to change the figure. Thus, my macros for including figures and tables causes the file name to be included in the printed output in small letters enclosed in small square brackets. When it comes time to create the final manuscript, I delete the instances of `\texttt{\small[#2 #3]}` from the macros and recompile the book's LaTeX files.

The definitions of `\snfig` and `\sntab` also include several lines that are commented out. Professional editors often like to see the manuscript with figures or tables each on its own page rather than in-line with the text. Commenting in these few lines puts the figures and tables of the whole book on their own pages.[3]

The `\snfig`, `\sntab`, and `\swsnfig` macros also define labels for cross referencing the figures with `\ref` or `\pageref` commands. A slight limitation of my implementation is that I cannot reuse the same figure or table file without confusing the labeling. However, it is easy enough to create a duplicate figure or table with a different file name.

I typically create all figures and most tables outside of TeX itself and include them from separate files. If I found myself inserting very many tables directly into my `.tex` files rather than including them from graphics files, I would define a `mytable` environment so that I could still contain and simply change the sort of formatting I have discussed.

2.7 Thought breaks

The next group of commands (mostly commented out) are various options for indicating what I call "thought breaks" — places where formatting indicates a change of topic big

3. The answer to the second question of the Ask Nelly column in this issue describes a different way to put each figure on its own page.

enough to highlight but not big enough to have its own section or subsection title. (These commands are defined with `\def` because I know they will pick up the correct arguments this way, and I am not sure enough of the details of how `\newcommand` works. (I understand the details of how TEX defines a macro and then collects its arguments when the macros are called because Knuth explains it pretty completely in *The TEXbook*. In particular, TEX allows macro calls where the arguments of the macro are not all embedded in pairs of braces. However, I have never stumbled across a rigorous explanation of how a macro defined with `\newcommand` collects its arguments and thus in what situations arguments not in braces will be recognized or to what extent LATEX defined macros can have both of what Knuth calls delimited and undelimited arguments — and I have not bothered to study the LATEX code to figure it out. Consequently, out of ignorance, I use `\def` to define macros which don't have their arguments delimited by braces.)

```
%\def\newthoughtgroup#1{\bgroup \afterassignment\BigFirstLetter \let\next=}
%\def\BigFirstLetter#1{\bigskip\noindex{\Large#1}}

%adapted slightly from Victor Eijkhout on c.t.t
%\def\newthoughtgroup#1{\BigFirstLetter#1$}
%\def\BigFirstLetter#1#2${\bigskip\noindent{\Large #1}#2}

\def\newthoughtgroup#1{%
   \bigskip\noindent {\large #1}}

%\def\newthoughtgroup{%
%  \bigskip\noindent }

%big bold dropped cap letter with rest of word small caps
%\def\newthoughtgroup#1#2 {\bigskip\noindent\lettrine{#1}{#2}\ }

%\def\thoughtbreak{\vskip2pt
%    \centerline{$^{\vrule width2cm height 1pt}$}\vskip2pt\noindent}
```

The version of `\mythoughtgroup` currently not commented out indicates the new thought by a vertical space and a slightly bigger capital letter at the beginning of a non-indented paragraph.

2.8 Chapter formatting

The final set of commands in my class file has to do with with the beginning and ends of chapters. At the beginning and ending of each chapter I insert some commands that I can change either by changing the commands themselves or changing macros in the class file.[4]

```
\RequirePackage{fancyhdr}
\pagestyle{fancyplain}
\newcommand{\mypartname}{}
\newcommand{\mychaptername}{}
\lhead[\fancyplain{}{\thepage}]{\fancyplain{}{}}
\chead[\fancyplain{}{\mypartname}]{\fancyplain{}{\mychaptername}}
\cfoot[\fancyplain{}{}]{\fancyplain{\thepage}{}}
\rhead[\fancyplain{}{}]{\fancyplain{}{\thepage}}
\newcommand{\EMPTYPAGE}{\clearpage\thispagestyle{empty}\cleardoublepage}

\newcommand{\ENDCHAPTER}{\dumpendnotes}
\newcommand{\fENDCHAPTER}{\vfil\dumpendnotes}
```

In the class file for the book from which I drew these illustrations, there are a couple of alternative macros that I can include at the end of each chapter to dump the endnotes, but the end-of-chapter macros could be defined to cause other actions and outputs. In this book (which has only 10 chapters) I do not combine everything in a single beginning-of-chapter macro (e.g., `\BEGINCHAPTER`), but I have done this with some books (e.g., the 20 chapter book I am also currently working on). The typical beginning-of-chapter commands for the chapter with the file name `rapid.tex` (mentioned earlier) are

```
\EMPTYPAGE
\chapter{Rapid Change in a Global World}
\label{ch:rapid}
\renewcommand{\mychaptername}{Chapter \thechapter: Rapid Change in a Global World}
```

2.9 Using a fully developed class

For some books I have included different or additional capabilities in my custom class file.

4. When I first started customizing my page headings a few years ago, I used the fancyheadings package; recently I have learned that the package fancyhdr has replaced fancyheadings, but I have not yet bothered to rewrite all the heading commands to use the new forms that come with the fancyhdr package and don't use the fancyplain device.

Obviously, I could also use a fully developed class such as memoir rather than making lots of modifications of my own to the LaTeX's standard book class. However, I suspect I would still use some of the ideas I have described above.

It is clear that TeX and its derivatives with their explicit, visible markup provide a strong base for incrementally building a personal library of techniques that are easy to apply from one project to the next.

Acknowledgements

I owe thanks to many sources for what I have learned about using LaTeX — books, the `comp.text.tex` list, the `texhax` list, and many individuals. Of course, none of them are responsible for lessons I have mislearned.

I can't remember and acknowledge everyone who directed me to techniques illustrated in this column; however I can remember some of them. Karl Berry reviewed an early version of this column and earlier told me about some of the methods I have described here. I also remember Peter Flynn, Steve Peter, and Steve Schwartz telling me about particular techniques. Peter Flom, Will Robertson, and the anonymous reviewers provided many helpful suggestions about the manuscript.

Biographical note

David Walden is retired after a career as an engineer, engineering manager, and general manager involved with research and development of computer and other high tech systems. More history is at `www.walden-family.com/dave`.