Organizing a large collection of stylefiles

Angelika BINDING

Springer Verlag Heidelberg, Dept. New Technologies / Product Development
Tiergartenstr. 17, W-6900 Heidelberg
e-mail: binding@dhdspri6.bitnet

Abstract. Springer Verlag has to maintain a large collection of macropackages for different layouts, for which there are versions both for plainTEXand LATEXand for different sets of fonts. We therefore designed a concept of modularising these packages and have implemented mechanisms to create formatfiles loading our individual set of fonts without changing the standard formatfiles plain and lplain.

Résumé. *Springer Verlag doit conserver et assurer la mise à jour d'un grand nombre d'ensembles de macros pour différentes maquettes, pour lesquels il existe une version plainTEX et une version LATEX et différentes polices de caractères. Nous avons donc développé l'idée de rendre ces ensembles de macros modulaires et nous avons implémenté les mécanismes permettant de créer des fichiers de format qui chargent nos propres polices sans avoir besoin de modifier les formats standards* plain *et* lplain.

Key words: plain-TEX, LATEX, fonts, formatfiles, layout

1. Introduction

Springer Verlag is a publishing company that publishes books and journals in medical and the natural sciences. When more and more authors switched to TEX to write their texts, Springer began writing macro packages to suit the various layouts that are used by our company.

Our publications can be divided into three categories: single-author or single-team books (monographs), multi-author books, consisting of contributions by various authors or teams but with a common editor (e.g. proceedings) and articles for journals. For our journals we have one-column and two-column formats. Fortunately, there is a standard layout for each of these, which together cover almost all our journals. In the single- and multi-author book categories however, there are a number of different layouts.

Things seemed quite harmless at the beginning. We created plain-TeX macro packages for camera-ready books, using cm fonts which were used via ordinary `\input` statement. But pretty soon we wanted to choose between cm fonts and postscript fonts. Then one of the phototypesetting companies we work with offered the possibility of typesetting TeX documents using the original Monotype Times fonts on their machine. They would also provide us with the necessary tfm files plus pk files for simulations of these fonts to allow previewing of the line and page breaks as they will appear in the final phototypeset version. A second company can now do the same thing, unfortunately with a slightly different set of fonts. So, even if we dispense with the old am fonts, this leaves us with three or four different font sets we have to integrate into our macro packages.

Also, we were quickly confronted with the need to support LaTeX and AMSTeX. Other TeX macro packages such as LAMSTeX, are to come. As all this will quickly lead to an explosion of the number of variations of a particular stylesfile we have to maintain, we had to think of a way of organizing things.

I explain here our concept regarding plain-TeX and LaTeX, disregarding AMSTeX for the moment.

2. plain-TeX

Hopefully, an author will publish more than one text with Springer, so textual elements should have the same macro names throughout all Springer plain-TeX packages. Appendix 1 gives you the list of all the frontend macros we have created.

As these macros work in the same way for most layouts, we parametrized them fully, extracting all layout-specific items of information and passing them through variables to the macro.

Let me demonstrate this with the macro for a heading of order 4 used in the single author package `psing01`:

```
% This is titd1.tex, containing the macro for a heading of order 4
\catcode`@=11    % use @ as a normal character
\@llocundefineddimen\tdgap\default{0.5em}
\def\titled#1{\par%
\if N\lasttitle% previous text was not a titlec
   \removelastskip
\else
```

```
   \vskip\tdbeforeback % else skip backwards
\fi
\bgroup
   \textfont0=\tdtt \scriptfont0=\tdts \scriptscriptfont0=\tdtss
   \textfont1=\tdmt \scriptfont1=\tdms \scriptscriptfont1=\tdmss
   \textfont2=\tdst \scriptfont2=\tdss \scriptscriptfont2=\tdsss
   \textfont3=\tdxt \scriptfont3=\tdxs \scriptscriptfont3=\tdxss
   \tdtt% select default font
   \vskip\tdbefore
   \if N\lasttitle \penalty\subsubsection@penalty \fi
   \global\subsubsection@penalty=-50
   \noindent
   \ignorespaces#1\unskip\kern\tdgap
\egroup
\ignorespaces}
\catcode`@=12    % reset catcode
```

The fonts are assigned in a dataset called `pfontnam.s01`:

```
...
\let  \tdms                 = \tbmss
\let  \tdmss                = \tamssss
\let  \tdmt                 = \tamss
\let  \tdss                 = \sevensy
\let  \tdsss                = \fivesy
\let  \tdst                 = \tensy
\let  \tdts                 = \sevenbf
\let  \tdtss                = \fivebf
\let  \tdtt                 = \tenbf
\let  \tdxs                 = \tenexatseven
\let  \tdxss                = \tenex
\let  \tdxt                 = \tenex
...
```

The sizes of the skips are specified in `pvalues.s01`

```
...
% \titled
\newskip\tdbeforeback    \tdbeforeback=-7pt % corrective space to \titlec
\newskip\tdbefore        \tdbefore=5.833pt plus 4pt minus4pt % space above
\newskip\tdgap           \tdgap=1em
...
```

We thus have a set of modules which are common to the different stylefiles. A stylefile merely specifies a number of fonts and size parameters and calls the modules needed.

This greatly simplifies maintenance. Occurrence of a bug or a change of layout doesn't mean updating dozens of macro packages; only one module has to be altered. Creation of new stylefiles is much faster than before.

3. Formatfiles

As our technical editors want to make available to the authors, all the fonts that are common in phototypesetting, loading all these in addition to the preloaded ones causes fontmemory problems on most installations.

A new formatfile is needed which loads only the necessary fonts and of course the macros. Simply editing the `plain.tex` or `lplain.tex` files will work for the moment, but a new version of these may make you have to start again from scratch. A more elegant way is to read in the original files and change their behavior "from the outside".

Here is our solution to this problem: The formatfile is created by running `initex` with a file like this one:

```
% initex file for single author package no. 01, as of 13.4.91
\input xplain.tex
\input hyphen.tex
%
\def\fontcm{cm}  % computer modern fonts
\def\fontmt{mt}  % Monotype Times fonts
\def\fontps{ps}  % postscript fonts
\def\fonttype{\fontcm}  % insert the desired fonttype here
%
\input pstart.s01       % setup for this package
\input pvalues.s01      % settings and measures
\input pfontsel.tex     % preliminary macros
\input pfont.tex        % get fonts
\input pfontnam.s01     % establish names in the package
...
\input ptitd1.tex       % \titled
\input ptitel.tex       % \titlee
%
\dump
```

This is the file `xplain.tex`.

```
% this is splain.tex as of 13.5.91
\catcode`\{=1 % left brace is begin-group character
\catcode`\}=2 % right brace is end-group character
```

```
\catcode'\#=6 % hash mark is macro parameter character
%
\let\qaya=\input \def\input#1 {\relax}
%
\let\qayb=\font
\def\font#1=#2
   {\def\qayba{#1}\def\qaybb{#2}
    \def\qaybf{\preloaded}
    \def\qaybg{\tenrm}\def\qaybh{\tenex}
    \def\qaybu{cmr7}\def\qaybv{cmtt10}\def\qaybw{cmssbx10}
    \ifx\qayba\qaybg \qayb#1=#2            % tenrm loaded
     \else\ifx\qayba\qaybh \qayb#1=#2      % tenex loaded
     \else\ifx\qayba\qaybf
     \ifx\qaybb\qaybu \qayb#1=#2           % cmr7 preloaded
     \else\ifx\qaybb\qaybv \qayb#1=#2      % cmtt10 preloaded
     \else\ifx\qaybb\qaybw \qayb#1=#2      % cmssbx10 preloaded
     \fi\fi\fi \fi\fi\fi}
%
\let\qayc=\skewchar \def\skewchar#1=#2 {\relax}
\let\qayd=\textfont \def\textfont#1=#2{\relax}
\let\qaye=\scriptfont \def\scriptfont#1=#2{\relax}
\let\qayf=\scriptscriptfont \def\scriptscriptfont#1=#2{\relax}
%
\qaya plain.tex
%
\let\input=\qaya \let\qaya=\undefined
\let\font=\qayb \let\qayb=\undefined
\let\skewchar=\qayc  \let\qayc=\undefined
\let\textfont=\qayd  \let\qayd=\undefined
\let\scriptfont=\qaye  \let\qaye=\undefined
\let\scriptscriptfont=\qayf  \let\qayf=\undefined
```

First three catcodes have to be defined. Next, "safety copies" are made of the commands `\input` and `\font`, plus a number of related commands. Except for `\font`, all these commands are then set to `\relax`. `\font` is redefined so that only those fonts will be loaded or preloaded which are absolutely necessary for `initex`. Then the original file `plain.tex` is read in using the safety copy of `\input`. Additional `\input` statements in this file will not be executed, and `\font` statements will only be executed for cmr10, cmex, cmr7, cmtt10 and cmssbx10. At the end all commands are restored to their original meaning and all safety copies are deleted.

Now, with unnecessary fonts cleared, we are free to include hyphenation patterns, fonts, "national" macros and layout macros as needed. As already mentioned, our macros must run with different sets of fonts. These fonts are listed in one file and a switch ensures that the right selection is being loaded.

4. LaTeX

For LaTeX we wanted to support the standard syntax as far as possible. Authors thus don't have to learn many new commands and texts that have already been written using standard LaTeX can be ported to our layout with only a few changes to the input.

Unfortunately, the layout changes we make involve macros that are defined partly in the styles and partly in the substyles. In some cases macros from `latex.tex` have to be redefined and in some instances completely new macros have to be created. In the past we achieved this by collecting all necessary elements in one dataset as our new style file. But here again, this is a quick and very dirty solution which will only work until the next major revision of LaTeX comes along. So, as in plain, we load original files and define changes to these afterwards.

Although it may seem bizarre at first glance, we always take the article stylefile as a basis. But by sticking to one stylefile we manage to be affected only by changes to this one stylefile, article being the style with the least features thus giving us the most flexibility.

We have to admit that we were not very modest in our wishes:

We wanted to make major changes to the standard LaTeXstyle article without changing the original files.

We wanted to be able to load different sets of fonts without loading unnecessary ones. We decided to use the new font selection scheme of Mittelbach and Schöpf for this purpose.

We wanted to be able to allow only certain options or exclude options altogether.

And, last but not least we wanted users to create formatfiles for our style and still use the standard LaTeX opening `\documentstyle{...}`. This calls for a safety mechanism that prevents specifiing `\documentstyle{A}` and running it with formatfile `B`.

But here is the code that does it all[1]

```
% Loading LaTeX / Fontsel environment
\catcode`\{=1 % left brace is begin-group character
\catcode`\}=2 % right brace is end-group character
```

[1] In reality this input is split into several datasets to maintain greater transparency.

```
\catcode'\#=6 % hash mark is macro parameter character
%
\let\tupni=\input
\def\input#1 {\def\tupnih{#1}\def\tupnij{hyphen}\def\tupnik{lfonts}
              \def\tupnil{fontdef.tex}\def\tupnim{preload.tex}
              \def\tupnin{xxxlfont.sty}
              \ifx\tupnih\tupnij \message{patterns loading below}
               \else\ifx\tupnih\tupnik \tupni lfonts.new
                \else\ifx\tupnih\tupnil \tupni fontdef.ori
                 \else\ifx\tupnih\tupnim \tupni preload.min
                  \else\ifx\tupnih\tupnin \tupni oldlfont.sty
                   \else \tupni #1\fi\fi\fi\fi\fi}
%
\tupni lplain.tex
%
\let\input=\tupni
\makeatletter
\let\@@input=\tupni
\def\input{\@ifnextchar \bgroup{\@iinput}{\@@input }}
\def\@iinput#1{\@@input #1 }
\makeatother
\let\tupni=\relax
%
\input hyphen

% Necessary utilities used in Springer Styles. Is necessary to load
% it before any stylefile modifications.
\makeatletter

\def\@optiondef#1{\expandafter\def\csname sp@#1\endcsname{}}

\def\@springererr#1#2{%
\edef\@tempc{#2}\expandafter\errhelp\expandafter{\@tempc}%
\typeout{Springer Style error.  \space See documentation for
 explanation.^^J \space\@spaces\@spaces\@spaces Type \space H <return>
 \space for immediate help.}\errmessage{#1}}
\def\springerstylefile{}

% Changes of the LaTeX start routines
\def\@documentstyle[#1]#2{\makeatletter
  \def\@tempx{#2}
  \def\@optionlist{#1}\gdef\@optionfiles{}%
  \@ifundefined{springerstylefile}{\input \@tempx.sty\relax}%
    {\ifx\@tempx\springerstylefile\relax \else
    \@springererr{Style '\springerstylefile' is preloaded.}{Please
     check your input.}\fi\@options}
    \let\@elt\input \@optionfiles \let\@elt\relax \makeatother\mark{{}{}}}

\def\@options{\let\@elt\relax
```

```
  \@for\@tempa:=\@optionlist\do{\@ifundefined{sp@\@tempa}%
    {\@springererr{Option '\@tempa' is not allowed.}
    {Please check your documentation.}}%
    {\@ifundefined{ds@\@tempa}{\xdef\@optionfiles{\@optionfiles
     \@elt \@tempa.sty\relax}}{\csname ds@\@tempa\endcsname}}}}
% Option checking

% Table of options:
% Use \@optiondef{NAME} to enable the option NAME.
%
\@optiondef{aa}
\@optiondef{bb}
\@optiondef{draft}
\@optiondef{twoside}
% Loading article stylefile and art12 substylefile
% Disable \@options before and enable it after all.
%
\@ifundefined{springerstyletest}{%
  \typeout{Loading article style and art12 substyle}
  \let\@tempy\@options \let\@options\relax
  \let\@tempi\input \def\input#1\relax{}
  \@tempi art12.sty
  \@tempi article.sty
  \let\input\@tempi
  \let\@options\@tempy}{\relax}
% Stylefile modifications start here:
\input fontdef.cm
\input preload.cm
\input fontcm.tex
\input symbols.tex
\input ucgreek.tex
\input vector.tex
\input schalter.s01
\input layout.s01
\input title.s01
\input runnhead.s01
\input theorem.s01
\input figure.s01
\input bib.s01
\input fonote.s01
\input acknow.s01
\input item.s01
\input typeset.s01
%
\makeatother
%
% define stylefile name
\def\springerstylefile{lsing01}
```

As with plain, we start by saving the `\input` command and redefining it into a filter that exchanges certain files that are called in `lplain` and skips inputting all the other files. After `lplain` is read in, the desired hyphenation patterns are loaded. This is followed by a change to the `\documentstyle` command to make it check for conflicting documentstyle or forbidden option specifications. When the base style article and an appropriate substyle (e.g. art12) have been loaded, things are finally ready for our font specifications and the actual layout commands. The file ends with the definition of the name of the stylefile.

5. Names of the stylefiles

In the process of modularizing our macro packages, we also designed a better names concept. The formatfiles now have 8-digit/letter names, made up according to the following rule.

The first letter gives the kind of fonts that are used:

c means Computer Modern fonts

m means Monotype Times fonts

p means Postscript fonts

The second letter tells which major TEXpackage is used with this formatfile:

p means plain-TEX

l means LATEX

a means AMSTEX

This leaves us 6 digits/letters to distinguish between the different layouts. Names of formatfiles for journals end with some kind of short form of the journals name. Formatfiles for single author formats have names of the form `...sing`*nn*, where *nn* is a two digit number. Multi-author formats have formatfiles denoted `...mult`*nn*.

6. Distribution

Our stylefiles are distributed free of charge either on diskette or, if delivery of font files is not necessary, as electronic mail. Distribution via fileserver is planned.

7. List of frontend macros

Explanation: *1-col* means 1-column formats for journals, *2-col* means 2-column formats for journals, *sing* stands for stylefiles for monographs and *mult* stands for stylefiles for multi-author books.

List of Springer macros	1-col	2-col	sing	mult
1. Title page				
`\maketitle`	*	*		
produces top matter using				
`\HEADNOTE`	*	*		
`\MAINTITLE`	*	*		
`\MAINTITLERUNNING`	*	*	*	
`\SUBTITLE`	*	*	*	
`\AUTHOR`	*	*	*	
`\AUTHORRUNNINGHEAD`	*	*	*	
`\INSTITUTE`	*	*	*	
`\DATE`	*	*		
`\SUMMARY`	*	*		
`\KEYWORDS`	*	*		
`\SUBCLASS`	*	*		
`\THESAURUS`	*	*		
`\SENDOFF`	*	*		
`\TRANSTITLE`	*	*		
`\TRANSSUM`	*	*		
`\TRANSKEY`	*	*		
`\DEDICATION`	*	*		
3. Doublespaced output for refereeing `\refereelayout`	*	*	*	*
4. Language setting				
`\english` or `\french` or `\german`; default: `\english`	*	*		
5. Front page of book				
`\bookhead`			*	*
`\bookauthor`			*	*

List of Springer macros	1-col	2-col	sing	mult
6. Headings				
`\part`			*	*
6.1 Numbered				
`\titlea`	*	*	*	*
`\titleb`	*	*	*	*
`\titlec`	*	*	*	*
6.2 Unnumbered				
`\utitlea`	*	*	*	*
`\utitleb`	*	*	*	*
`\utitlec`	*	*	*	*
`\utitled`	*	*	*	*
`\utitlee`	*	*	*	*
`\utitlef`	*	*	*	*
6.3 Changing running titles				
`\titlearunning`			*	*
`\titlebrunning`			*	*
7. Various headings				
`\intro`			*	
`\foreword`			*	
`\preface`			*	
`\dedication`			*	
8. Motto				
`\motto`			*	*
9. Table of contents				
`\contents`			*	*
`\contpart`			*	*
`\contcontribution`			*	
`\conttitlea-c`			*	*
10. Index				
`\makeindex`			*	*
`\Idx`, `\Asidx`			*	*
`\printindex`			*	*
11. Small print				
`\begpet ... \endpet`	*	*	*	*

List of Springer macros	1-col	2-col	sing	mult
12. Figures				
ordinary: `\begfig`	*	*	*	*
two figure side by side:				
`\begdoublefig`		*	*	*
one figure, legend at the side:				
`\begfigside`	*	*	*	*
two column figure:				
`\begfigwid`		*		
Legend: `\figure`	*	*	*	*
13. Tables				
Paste-in tables:				
`\begtabempty`	*	*	*	*
Paste in, two columns:				
`\begtabemptywid`		*	*	*
TeXed tables:				
`\begtabfull`	*	*	*	*
TeXed tables, two columns:				
`\begtabfullwid`		*	*	*
Table captions: `\tabcap`	*	*	*	*
14. Empty pages				
`\freepage`			*	*
15. References				
15.1 Headings:				
`\begref`	*	*	*	*
`\begrefbook`			*	*
`\begrefchapter`			*	*
`\nextchapter`			*	*
15.2 Entries				
Unnumbered: `\ref`	*	*	*	*
Numbered: `\refno`	*	*	*	*
Letter system: `\refmark`	*	*	*	*
15.3 End of References				
`\endref`	*	*	*	*

List of Springer macros	1-col	2-col	sing	mult
16. Acknowledgements				
`\acknow`	*	*	*	*
17. Appendices				
`\app`	*	*	*	*
18. Note added in proof				
`\noteadded`	*	*		
19. Footnotes				
Ordinary: `\fonote`	*	*	*	*
Starred ones for maintitles:				
`\FOOTNOTE`	*	*		
For present address:				
`\PRESADD`	*	*		
20. Minienvironments				
`\beglemma ... \endlemma`	*	*	*	*
dito for:				
definition	*	*	*	*
theorem	*	*	*	*
corollary	*	*	*	*
proof	*	*	*	*
example	*	*	*	*
property	*	*	*	*
conjecture	*	*	*	*
claim	*	*	*	*
remark	*	*	*	*
20.1 Defining new environments				
numbered: `\devenva`	*	*	*	*
unnumbered: `\devenvb`	*	*	*	*

List of Springer macros	1-col	2-col	sing	mult
21. End of document				
`\bye`	*	*	*	*
22. Mathematical symbols				
Vectors: `\vec`	*	*	*	*
Tensors: `\tens`	*	*	*	*
Fraktur symbols: `\frak`	*	*	*	*
Script symbols: `\cal`	*	*	*	*
Various symbols	*	*	*	*
23. Frames: `\frames`			*	*