`psfont`: A general way to use PostScript fonts[*]

Sebastian Marius Kirsch
`sebastian_kirsch@kl.maus.de`

1996/06/30

Abstract

This package provides a general way to use PostScript fonts without handling with thousands of style files, one for each font you want to use. It works with a style file, `psfont.sty`, which contains all the macros, and a configuration file, `psfont.cfg`, which tells the style file about the available fonts.

1 Introduction

This package lets you use one single package to load every PostScript font you would want. You don't have to load one file for each font anymore, and the package even warns you if you try to load two contradicting fonts at once, e. g. two roman fonts. If you use the option `onlyps`, it substitutes the default PS fonts for the families you have not specified.

The package is intended as a replacement for the psfonts package of psNFSS, by Sebastian Rahtz (`s.rahtz@elsevier.co.uk`). psfonts is not very consequent, because some of its style files redefine all three defaults, while others only change one.

2 The `docstrip` modules

This file contains five modules to direct `docstrip` in generating the external files:

driver	A short driver for producing the documentation
package	The package itself
config	The local configuration file
Upsy	A font definition file for Adobe Symbol, from psNFSS
Upzd	A font definition file for Adobe Zapf Dingbats, from psNFSS

3 Producing the documentation

This short driver can be extracted by `docstrip` to produce the documentation.

```
⟨*driver⟩
```

---

[*] This file has version number 1.1, last revised 1996/06/30.

```
\documentclass{ltxdoc}

\newcommand{\NFSS}{\textsf{NFSS}}
\newcommand{\psNFSS}{\textsf{psNFSS}}
\newcommand{\psfont}{\texttt{psfont}}

\begin{document}

\DocInput{psfont.dtx}

\end{document}
⟨/driver⟩
```

4 The Code

4.1 Introduction

First we have to introduce ourselves.

```
⟨*package⟩
\NeedsTeXFormat{LaTeX2e}
\ProvidesPackage{psfont}%
  [\filedate\space v\fileversion\space loading PostScript fonts]
```

4.2 Switches

Next, we define some switches. They are used to determine whether some fonts have already been loaded, so we can warn the user when he tries to load two colliding fonts.

```
\newif\ifrm@set\rm@setfalse
\newif\ifsf@set\sf@setfalse
\newif\iftt@set\tt@setfalse
```

4.3 The default fonts

`\DefaultRMFont`
`\DefaultSFFont`
`\DefaultTTFont`

These are some commands for changing the default fonts.
`\DefaultRMFont{`⟨*NFSS font family*⟩`}`
`\DefaultSFFont{`⟨*NFSS font family*⟩`}`
`\DefaultTTFont{`⟨*NFSS font family*⟩`}`

```
\newcommand{\DefaultRMFont}[1]{\renewcommand{\def@rmfont}{#1}}
\newcommand{\DefaultSFFont}[1]{\renewcommand{\def@sffont}{#1}}
\newcommand{\DefaultTTFont}[1]{\renewcommand{\def@ttfont}{#1}}
\newcommand{\def@rmfont}{}
\newcommand{\def@sffont}{}
\newcommand{\def@ttfont}{}
```

4.4 Declaring the options

`\AvailableRMFont`
`\AvailableRMFont`
`\AvailableRMFont`

These commands are needed later in the configuration file. There is one command for each font family, ie. roman, sans serif or typewriter.

`\AvailableRMFont[`⟨*additional code*⟩`]{`⟨*long name*⟩`}{`⟨*NFSS family name*⟩`}`
`\AvailableSFFont[`⟨*additional code*⟩`]{`⟨*long name*⟩`}{`⟨*NFSS family name*⟩`}`
`\AvailableTTFont[`⟨*additional code*⟩`]{`⟨*long name*⟩`}{`⟨*NFSS family name*⟩`}`

They declare their first argument as an option.

If the option is called, and the font family has already been defined, a `\PackageError` is reported. If not, the second argument is defined as the approriate family default.

When the family default has successfully been set, the approriate switch is turned, so it is not redefined anywhere else in this file.

The optional argument contains some code that is to be executed when the defaults have been set. I do not need this now, but this hook could be useful later.

```
\newcommand{\AvailableRMFont}[3][]{%
  \DeclareOption{#2}{%
    \ifrm@set\PackageError{psfont}{%
      \protect\rmfamily\space already defined as \rmdefault
      }{%
      You tried to load two roman families at the same time,\MessageBreak
      e.g. times and palatino}
    \else\renewcommand{\rmdefault}{#3}\rm@settrue
    #1
    \fi}
  }

\newcommand{\AvailableSFFont}[3][]{%
  \DeclareOption{#2}{%
    \ifsf@set\PackageError{psfont}{%
      \protect\sffamily\space already defined as \sfdefault
      }{%
     You tried to load two sans serif families at the same time,\MessageBreak
      e.g. gill and helvetica}
    \else\renewcommand{\sfdefault}{#3}\sf@settrue
    #1
    \fi}
  }

\newcommand{\AvailableTTFont}[3][]{%
  \DeclareOption{#2}{%
    \iftt@set\PackageError{psfont}{%
      \protect\ttfamily\space already defined as \ttdefault
      }{%
     You tried to load two typewriter families at the same time,\MessageBreak
      e.g. courier and typewriter}
    \else\renewcommand{\ttdefault}{#3}\tt@settrue
    #1
    \fi}
  }
```

4.5 One command fits all

`\AvailableFont` With the macro `\AvailableFont`, all three defaults are set with one option. This is useful for managing combinations of three fonts under one name, as well as for fonts that have the variants sans serif and typewriter. (I'm told Lucida is one of those.)

```
\newcommand{\AvailableFont}[5][]{%
  \DeclareOption{#2}{%
    \ifrm@set\PackageError{psfont}{%
      \protect\rmfamily\space already defined as \rmdefault
      }{%
      You tried to load two roman families at the same time,\MessageBreak
      e.g. times and palatino}
    \else\renewcommand{\rmdefault}{#3}\rm@settrue
    \fi
    \ifsf@set\PackageError{psfont}{%
      \protect\sffamily\space already defined as \sfdefault
      }{%
     You tried to load two sans serif families at the same time,\MessageBreak
      e.g. gill and helvetica}
    \else\renewcommand{\sfdefault}{#4}\sf@settrue
    \fi
    \iftt@set\PackageError{psfont}{%
      \protect\ttfamily\space already defined as \ttdefault
      }{%
     You tried to load two typewriter families at the same time,\MessageBreak
      e.g. courier and typewriter}
    \else\renewcommand{\ttdefault}{#5}\tt@settrue
    \fi
    #1
  }
}
```

4.6 Loading the configuration file

Next, the configuration file is loaded. If it is not found, an error is issued, because the package is quite useless without it.

```
\InputIfFileExists{psfont.cfg}{}{%
  \PackageError{psfont}{%
    No local configuration file found
    }{%
    The psfont package was loaded without a local\MessageBreak
    configuration file, so it doesn’t know which fonts\MessageBreak
    are available.}
  }
⟨/package⟩
```

4.7 The configuration file

The configuration file looks like this:

4.7.1 Introduction

As usual…

```
⟨*config⟩
\NeedsTeXFormat{LaTeX2e}
\ProvidesFile{psfont.cfg}
  [\filedate\space v\fileversion\space Local configuration for psfont.sty]
```

4.7.2 Declaring default fonts

Here, the default PS fonts are defined. They are needed later, if the author does not define all PS fonts he wants to use, but wants his document to contain only PS fonts.

```
\DefaultRMFont{ptm}
\DefaultSFFont{phv}
\DefaultTTFont{pcr}
```

4.7.3 Available fonts

Now the available PS fonts are defined. They are divided into three categories, roman fonts, sans serif fonts and typewriter fonts, as it is done in $\LaTeX\,2_\varepsilon$ itself.

They are defined using the `\Available`... macros. These macros define a symbolic name to be used as an option to the package, and take the name of the NFSS font family as a second argument.

```
\AvailableRMFont{times}{ptm}
\AvailableRMFont{palatino}{ppl}
\AvailableRMFont{newcent}{pnc}

\AvailableSFFont{helv}{phv}
\AvailableSFFont{gill}{pgs}

\AvailableTTFont{courier}{pcr}
⟨/config⟩
```

And that's the configuration. Of course one could do much more with this configuration file, declaring new options and all. It's probably the biggest hook a package could have. :-)

4.8 `onlyps`: Using only PS fonts

`onlyps` This option redefines all other family defaults that have not yet been `\renewcommand`ed to PS fonts. It uses the default fonts as defined in the configuration file. If no default fonts have been defined, it reports a warning and does nothing.

```
⟨*package⟩
\DeclareOption{onlyps}{
  \ifrm@set\else
    \ifx\empty\def@rmfont
      \PackageWarning{psfont}{No default roman font defined!}
    \else\renewcommand{\rmdefault}{\def@rmfont}\fi
  \fi
  \ifsf@set\else
    \ifx\empty\def@sffont
      \PackageWarning{psfont}{No default sans serif font defined!}
    \else\renewcommand{\sfdefault}{\def@sffont}\fi
  \fi
  \iftt@set\else
    \ifx\empty\def@ttfont
      \PackageWarning{psfont}{No default typewriter font defined!}
    \else\renewcommand{\ttdefault}{\def@ttfont}\fi
  \fi
  }
```

4.9 Parts of psfonts

The `psfont` *package ist ultimately intended as a replacement for psfonts of psNFSS. I therefore use the code of some files of psfonts and wrap it up in an option. All code and comments of this section are from psfonts, v5.2 by Sebastian Rahtz. Many thanks to Sebastian for letting me use his code!*

`pifont`

```
\DeclareOption{pifont}{
```

Now some useful commands for Pi fonts (Dingbats, Symbol etc); they all assume you know the character number of the (unmapped) font

```
\newcommand{\Pifont}[1]{\fontfamily{#1}\fontencoding{U}%
\fontseries{m}\fontshape{n}\selectfont}
\newcommand{\Pisymbol}[2]{{\Pifont{#1}\char#2}}
\newcommand{\Pifill}[2]{\leaders\hbox{\makebox[0.2in]{%
       \Pisymbol{#1}{#2}}}\hfill\kern\z@}
\newcommand{\Piline}[2]{\par\noindent\hspace{0.5in}\Pifill{#1}{#2}%
       \hspace{0.5in}\kern\z@\par}
\newenvironment{Pilist}[2]%
{\begin{list}{\Pisymbol{#1}{#2}}{}}%
{\end{list}}%
```

A Pi number generator (from ideas by David Carlisle), for use in lists where items are suffixed by symbols taken in sequence from a Pi font. Usage is in lists just like enumerate.

`\Pinumber` outputs the appropriate symbol, where `#2` is the name of a LaTeX counter and `#1` is the font family.

```
\def\Pinumber#1#2{\protect\Pisymbol{#1}{\arabic{#2}}}
\newenvironment{Piautolist}[2]{%
\ifnum \@enumdepth >3 \@toodeep\else
     \advance\@enumdepth \@ne
```

We force the labels and cross-references into a very plain style (eg no brackets around ‘numbers’, or dots after them).

```
      \edef\@enumctr{enum\romannumeral\the\@enumdepth}%
  \expandafter\def\csname p@enum\romannumeral\the\@enumdepth\endcsname{}%
  \expandafter\def\csname labelenum\romannumeral\the\@enumdepth\endcsname{%
     \csname theenum\romannumeral\the\@enumdepth\endcsname}%
  \expandafter\def\csname theenum\romannumeral\the\@enumdepth\endcsname{%
     \Pinumber{#1}{enum\romannumeral\the\@enumdepth}}%
  \list{\csname label\@enumctr\endcsname}{%
        \@nmbrlisttrue
        \def\@listctr{\@enumctr}%
        \setcounter{\@enumctr}{#2}%
        \addtocounter{\@enumctr}{-1}%
        \def\makelabel##1{\hss\llap{##1}}}
\fi
}{\endlist}
```

All the old Dingbat commands still work.

```
\newcommand{\ding}{\Pisymbol{pzd}}
\def\dingfill#1{\leaders\hbox{\makebox[0.2in]{\Pisymbol{pzd}{#1}}}\hfill}
\def\dingline#1{\Piline{pzd}{#1}}
\newenvironment{dinglist}[1]{\begin{Pilist}{pzd}{#1}}%
  {\end{Pilist}}
```

```
\newenvironment{dingautolist}[1]{\begin{Piautolist}{pzd}{#1}}%
  {\end{Piautolist}}
{\Pifont{pzd}}
{\Pifont{psy}}
}
```

`mathptm`

```
\DeclareOption{mathptm}{
```

This package loads the Adobe Times fonts and the mathptm fonts; The virtual fonts are produced by fontinst; they can be built by running tex on fontptcm.tex from the fontinst package.

```
% The main text family is Times Roman
\def\rmdefault{ptm}
\DeclareSymbolFont{operators}   {OT1}{ptmcm}{m}{n}
\DeclareSymbolFont{letters}     {OML}{ptmcm}{m}{it}
\DeclareSymbolFont{symbols}     {OMS}{pzccm}{m}{n}
\DeclareSymbolFont{largesymbols}{OMX}{psycm}{m}{n}
\DeclareSymbolFont{bold}        {OT1}{ptm}{bx}{n}
\DeclareSymbolFont{italic}      {OT1}{ptm}{m}{it}
```

If we're in compatibility mode, defined `\mathbf` and `\mathit`.

```
\@ifundefined{mathbf}{}{\DeclareMathAlphabet{\mathbf}{OT1}{ptm}{bx}{n}}
\@ifundefined{mathit}{}{\DeclareMathAlphabet{\mathit}{OT1}{ptm}{m}{it}}
```

An `\omicron` command, to fill the gap.

```
\DeclareMathSymbol{\omicron}{0}{operators}{`\o}
```

Reduce the space around math operators

```
\thinmuskip=2mu
\medmuskip=2.5mu plus 1mu minus 1mu
\thickmuskip=4mu plus 1.5mu minus 1mu
```

No bold math.

```
\def\boldmath{%
   \@warning{there is no bold Symbol font}%
   \global\let\boldmath=\relax
}
\DeclareMathSizes{5}{5}{5}{5}
\DeclareMathSizes{6}{6}{5}{5}
\DeclareMathSizes{7}{7}{5}{5}
\DeclareMathSizes{8}{8}{6}{5}
\DeclareMathSizes{9}{9}{7}{5}
\DeclareMathSizes{10}{10}{7.4}{6}
\DeclareMathSizes{10.95}{10.95}{8}{6}
\DeclareMathSizes{12}{12}{9}{7}
\DeclareMathSizes{14.4}{14.4}{10.95}{8}
\DeclareMathSizes{17.28}{17.28}{12}{10}
\DeclareMathSizes{20.74}{20.74}{14.4}{12}
\DeclareMathSizes{24.88}{24.88}{17.28}{14.4}
}
```

4.10 The end

At last, the options are processed.

```
\ProcessOptions
```

```
⟨/package⟩
```

5 Additional font definitions from **psfonts**

This is again code from psfonts *by Sebastian Rahtz. It contains some font definitions for Adobe Symbol and Adobe Zapf Dingbats.*

5.1 Adobe Symbol font

```
⟨*Upsy⟩
\typeout{File \space Upsy.fd\space loading \space Adobe\space Symbol}%
\DeclareFontFamily{U}{psy}{}%
\DeclareFontShape{U}{psy}{m}{n}{<->psyr}{}%
\DeclareFontShape{U}{psy}{m}{i}{<->ssub * psy/m/n}{}%
⟨/Upsy⟩
```

5.2 Adobe Zapf Dingbats

```
⟨*Upzd⟩
\typeout{File \space Upzd.fd\space loading \space Adobe\space ZapfDingbats}%
\DeclareFontFamily{U}{pzd}{}%
\DeclareFontShape{U}{pzd}{m}{n}{<->pzdr}{}%
⟨/Upzd⟩
```

6 ToDo

There is mainly one things that this package is missing till now:

- If the package has been loaded without any local options, it should act as if the option `onlyps` had been called. I unfortunately do not know how to detect this.

7 Legal rubbish