

ttf2tex

A TrueType Font Installer for Unix

Version 0.70

September 30, 2004

End-of-Life Announcement

As of September 30, 2004, `ttf2tex` is no longer maintained or in any way supported by its original author. Active development of the script has in fact already stopped in September 2002. Be advised that, even though `ttf2tex` may still work, there is no support in case of a problem and there will be no further updates.

Contents

1	Introduction	1
2	License	2
3	Requirements	2
4	Synopsis	4
5	Usage	5
6	Using the fonts	7
7	Tutorial	9
8	Hints and limitations	11
9	Acknowledgments	14

1 Introduction

`ttf2tex` is a script for the Bash shell which generates all files required to use TrueType fonts with `teTeX` from a set of font files. In short, it will do for TrueType fonts what `fontinst`'s `\latinfamily` command does for Type 1 PostScript fonts. In addition to that, `ttf2tex` sorts all files, builds the map files required by `ttf2pk` and `pdfTeX`, and optionally installs everything into either the system-wide local TeX tree or the private TeX tree of the user running `ttf2tex`. Note that `ttf2tex`'s approach to using TrueType fonts with TeX does *not* imply converting them to

Type 1 format. With a current version of `teTeX` and a proper setup, `TeX` can use TrueType fonts via the `ttf2pk` utility while `pdfTeX` even supports them natively.

`TeX` itself is actually completely indifferent to the font format since it will merely use the font metrics without accessing the glyph outlines. `ttf2pk` comes into play when processing the DVI file with utilities that do not support TrueType fonts natively, such as `dvips` or DVI readers like `xdvi`. `ttf2pk` will then render the glyphs and provide the result in PK font format. `pdfTeX`, on the other hand, has to access the fonts directly when embedding them in a PDF file. Fortunately it offers native TrueType font support and `ttf2tex` will supply it with a map file. Please see section 8.1 for a discussion of the limitations of this approach.

2 License

`ttf2tex` is copyright © 2000–2002, 2004 Philipp Lehman. This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 2, or (at your option) any later version.

This program is distributed in the hope that it will be useful, but *without any warranty*; without even the implied warranty of *merchantability* or *fitness for a particular purpose*. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with this program; if not, write to the Free Software Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301, USA.

3 Requirements

First of all, you need the Bash shell to run `ttf2tex`. Ash or plain sh will not work since `ttf2tex` uses Bash-specific extensions. `ttf2tex` depends on the utilities `vptovf`, `ttf2afm`, and `ttf2tfm`. Both `vptovf` and `ttf2afm` ship with `teTeX` and should be available on your system. `ttf2tfm` is part of the tools that come with an excellent TrueType rendering engine called FreeType. You need version 1.3.x or 1.4 of FreeType. 2.x versions will not work because `ttf2tfm` and `ttf2pk` have not been ported to FreeType 2 yet. Your Linux distributor or Unix vendor probably provides precompiled FreeType packages. Please consult the canonical sources for your distribution. I'm quite sure that at least all major Linux distributions come with FreeType packages.

Note that you need both the core library and the contributed binaries which might be distributed separately, so make sure you look for names like 'freetype' as well as 'freetype-contrib' or 'freetype-tools'. If your distributor or vendor does not provide FreeType packages you can download¹ the source code. You need either the package `freetype-1.3.1`² from the stable branch or the two packages `freetype-current`³ and `freetype1-contrib-current`⁴ from the unstable branch. Note that you have to install the FreeType tools anyway, since you need `ttf2pk` in order to actually use TrueType fonts with `TeX` and `LaTeX`, even though `ttf2tex` doesn't depend on `ttf2pk` directly.

¹<http://www.freetype.org/download.html>

²<ftp://ftp.freetype.org/freetype/freetype1/freetype-1.3.1.tar.gz>

³<ftp://ftp.freetype.org/freetype/unstable/freetype-current.tar.gz>

⁴<ftp://ftp.freetype.org/freetype/unstable/freetype1-contrib-current.tar.gz>

There are a few more things you need to do before running `ttf2tex` for the first time. If you haven't already done so, set up a local TeX tree `$TEXMFLOCAL` and a user TeX tree `$HOMETEXMF` in the global configuration file for `kpathsea`, `texmf.cnf`. Then run:

```
kpsexpand \TEXMFLOCAL
kpsexpand \HOMETEXMF
kpsexpand \TEXMF
```

to verify that these trees are set up properly and included in `$TEXMF`. Also make sure that `$TTFONTS` is set in `texmf.cnf`. To verify that, run: `kpsexpand \TTFONTS`. As to the FreeType tools, make sure that `ttf2pk` and `ttf2tfm` were built with `kpathsea` support. You can verify that by running them with the `--version` option:

```
ttf2pk --version
ttf2tfm --version
```

The output should include the `kpathsea` version used:

```
ttf2pk version 1.4 (kpathsea version 3.3.7)
ttf2tfm version 1.4 (kpathsea version 3.3.7)
```

If it does not, try to find binaries with `kpathsea` support or build the FreeType tools from source. You might also need to set `$TTF2PKINPUTS` in `texmf.cnf` to enable `ttf2pk` to locate its map file, `ttfonts.map`. To find out if doing so is necessary, run: `ttf2pk -t test`. `ttf2pk` should then print the following error message:

```
ttf2pk: ERROR: Cannot find font test in ttfonts.map.
```

This error message implies that `ttf2pk` managed to locate its map file. The actual error does not matter at this point since we merely want to make sure that `ttf2pk` is able to find `ttfonts.map`. `ttf2pk` might also exit with the following error message:

```
ttf2pk: ERROR: Cannot open file ttfonts.map.
```

This means that `ttf2pk` did not find its map file at all. In this case you should locate the file `ttfonts.map` on your system and define `$TTF2PKINPUTS` in `texmf.cnf`. Refer to the documentation of the FreeType tools for details. Note that the file searching mechanism of both `ttf2pk` and `ttf2tfm` is explained in the manual page of `ttf2tfm`. For most systems, the following settings should be fine:

```
TTF2PKINPUTS = .;$TEXMF/ttf2pk//
TTF2TFMINPUTS = .;$TEXMF/ttf2tfm//
```

After that, open the configuration file `ttf2tex.cfg` in a text editor and verify that all the paths and file names correspond to your installation. Read the comments given in the file for details. Make sure that there will be exactly one map file for `ttf2pk` on your system since, unlike `dvips` or `pdfTeX`, `ttf2pk` only supports a single map file called `ttfonts.map`. The location of the map file created or updated by `ttf2tex` is given in the script, make sure that it points to `ttfonts.map` if this file already exists. Create symbolic links to resolve any ambiguities if necessary. It is safe to have `ttf2tex` use an existing map file or a symbolic link pointing to it since it will only append data to the file. `ttf2tex` will not overwrite it.

4 Synopsis

```
ttf2tex.sh [options] --install --foundry <foundry> --font <font> <family>
```

The `--install` or `-i` option will create all files, install them, and append all mapping information to your map files using the system-wide local TeX tree. You may use the options in any arbitrary order and mix their long and short forms, but they cannot be concatenated. Options and arguments are as follows: Mandatory. Use the string *<foundry>* when creating subdirectories. The type foundry is the company which created the font, for example ‘monotype’. The string *<foundry>* must not contain any spaces.

<code>--foundry <i><foundry></i></code>	Mandatory. Use the string <i><foundry></i> when creating subdirectories. The type foundry is the company which created the font, for example ‘monotype’. This string must not contain any spaces.
<code>-f <i><foundry></i></code>	
<code>--font <i></i></code>	Mandatory. Use the string <i></i> when creating subdirectories. This string must not contain any spaces.
<code>-o <i></i></code>	Mandatory. The identifier of the font family, for example <i>times</i> or <i>mns</i> . The string <i><family></i> forms the basis of all file names and is used to identify the font when selecting it under LaTeX. It must not contain any spaces.
<code><family></code>	
<code>--batch</code>	Run in batch mode. By default, <code>ttf2tex</code> will ask for confirmation before processing the fonts. Batch mode skips the confirmation prompt. Please note that <code>ttf2tex</code> itself does not support batch processing. This switch is provided for cases in which it is run from a wrapper script.
<code>-b</code>	
<code>--log</code>	Write a transcript of your entire <code>ttf2tex</code> session to the file <code>ttf2tex.log</code> in the working directory. By default, <code>ttf2tex</code> will not create a log file.
<code>-l</code>	
<code>--typewriter</code>	Treat the font as a typewriter typeface. This option will disable hyphenation for the font by adding a declaration to the font definition file.
<code>-t</code>	
<code>--expert</code>	Run in expert mode. Use this option to install OpenType fonts which offer additional glyphs such as real small caps and old style figures <i>and</i> provide valid PostScript glyph names. It will implicitly activate <code>--ps-names-only</code> if required, <code>--caps</code> will be ignored. See the description of the <code>--ps-names-only</code> option and section 6.2 for details.
<code>-x</code>	
<code>--overwrite</code>	Run in overwrite mode. If this option is used, <code>ttf2tex</code> will overwrite any existing destination files when installing files into the local or private TeX tree. Use with care! Note that the map file for <code>ttf2pk</code> will not be overwritten. If neither <code>--install</code> nor <code>--user</code> is used, this option has no effect.

<code>--ps-names</code>	Pass the <code>-n</code> switch to <code>ttf2tfm</code> . This tells <code>ttf2tfm</code>
<code>-n</code>	to use the PostScript glyph names as given in the font file, but only if a glyph has a valid entry in the active cmap as well. See the <code>ttf2tfm</code> documentation for details.
<code>--ps-names-only</code>	Pass the <code>-N</code> switch to <code>ttf2tfm</code> . This tells <code>ttf2tfm</code> to
<code>-N</code>	always use the PostScript glyph names as given in the font file and ignore the font's cmaps. See the <code>ttf2tfm</code> documentation for details.
<code>--caps <real></code>	Use the real number <code><real></code> to determine the height of ('faked') small caps. The small caps will be <code><real></code>
<code>-c <real></code>	times the height of uppercase glyphs. A value of 0.8 for example will create small caps which are 80% the height of uppercase letters. <code>--help</code> reports the default value. This option will be ignored if <code>--expert</code> is used.
<code>--slant <real></code>	Use the real number <code><real></code> as obliqueness factor when generating slanted shapes. If this value is larger than zero, the characters slope to the right, otherwise to the left. This number is the tangent of the slant angle, it is usually much smaller than 1. A common choice is 0.167 (about 9.5°). <code>--help</code> reports the default value.
<code>-s <real></code>	

`ttf2tex.sh [options] --user --foundry <foundry> --font <family>`

The `--user` or `-u` option will create all the files, install them, and append all mapping information to your map files using the private TeX tree of the user running `ttf2tex`. Options and arguments as above.

`ttf2tex.sh [options] --foundry <foundry> --font <family>`

When omitting both `--install` and `--user`, `ttf2tex` will build a branch of your TeX tree in the current working directory and allow you to install the files later. Map files will be put into the working branch of `ttf2tex` as well, no file outside the working directory and its subdirectories will be touched. Options and arguments as above.

`ttf2tex.sh --dump-vectors`

Dump all internal encoding vectors to the working directory and exit. Any other options given on the command line will be ignored.

`ttf2tex.sh --help`

Print a brief usage summary including the defaults for `--caps` and `--slant` and exit. Any other options given on the command line will be ignored.

5 Usage

5.1 Renaming the font files

Before invoking `ttf2tex`, you need to rename the font files in a way that allows `ttf2tex` to guess the weight and the variant of the font by the name of the file.

Weight	Code	File name	
		Upright	Italic
ultra light, thin, hairline	a	fama16.ttf	famai16.ttf
extra light	j	famj16.ttf	famji16.ttf
light	l	faml16.ttf	famli16.ttf
book	k	famk16.ttf	famki16.ttf
regular	r	famr16.ttf	famri16.ttf
medium	m	famm16.ttf	fammi16.ttf
demibold	d	famd16.ttf	famdi16.ttf
semibold	s	fams16.ttf	famsi16.ttf
bold	b	famb16.ttf	fambi16.ttf
extra bold	x	famx16.ttf	famxi16.ttf
heavy	h	famh16.ttf	famhi16.ttf
black	c	famc16.ttf	famci16.ttf
ultra bold	u	famu16.ttf	famui16.ttf
poster	p	famp16.ttf	fampi16.ttf

Table 2: Weights and file names supported by `ttf2tex`

`ttf2tex` will search the working directory for all of the files listed in table 2. These file names are based on the following pattern: `fam` is the identifier of the whole font family. This string will be used when calling `ttf2tex` and when selecting the typeface under the new font selection system (NFSS) later. The next letter represents the weight of the font. For most text fonts you will only need `r` and `b` while some font families come with `k` and `d` instead. The next letter, `i`, indicates an italic font and is omitted for upright shapes. `16` is a fixed string which indicates Unicode encoding. The ‘16’ is required in the file names although it is in fact a mere matter of naming conventions and not actually used by `ttf2tex` to determine the encoding. `ttf2tex` always assumes Unicode encoding.

The weight codes used by `ttf2tex` are based on the Fontname scheme, a canonical naming system for font files which is also known by the name of its creator as the Karl Berry scheme. `ttf2tex` follows this scheme to a certain extend. Technically, canonical font naming is by no means required to use the fonts with a single TeX installation, although it is always a good idea when dealing with large numbers of fonts. If you are not interested in canonical naming you will find all you need to know in order to use `ttf2tex` in table 2.

5.2 Canonical font naming

There is a fairly elaborate canonical naming scheme used for the Type 1 fonts that come with `teTeX` as well as by `fontinst`, the TeX Type 1 font installation utility. You can read the documentation of the Fontname naming system online⁵ or download it as a tarball⁶ from any CTAN FTP mirror. See the file `fontname.dvi` for an overview as well as excerpts from various `map` files and browse the `.map` files for the complete listings. Of course these lists can’t cover all the fonts out there, so you might still need to create your own identifiers. Note that the way `ttf2tex` handles fonts differs slightly from what is described there. `ttf2tex` will

⁵<http://www.ctan.org/tex-archive/info/fontname>

⁶<ftp://tug.ctan.org/tex-archive/info/fontname.tar.gz>

only recognize the two variants ‘upright’ and ‘italic’ when parsing the file names and it does not recognize any widths (e.g. ‘condensed’ or ‘narrow’) at all. The Fontname naming system as described in the document mentioned above is based on the pattern:

```
S TT W [V...] [NN] [E] [DD]
```

S indicates the supplier (foundry), TT the typeface (two letters), W the weight, V the variant(s) (one or more letters), NN the encoding (usually two letters), E the expansion (width) and DD the design size. Square brackets indicate an italic font and omitted for upright shapes. `ttf2tex` however can only handle the pattern:

```
S TT... W [V] 16
```

That is, the design size is omitted (it is not used with linearly scalable fonts anyway, even in the Fontname scheme) and the expansion as well as all variants other than italic have to become part of the font family name which is not limited to two characters. If you want to name the font files according to the canonical scheme, you have to take this into account – and can only follow the scheme to a certain extend, although you probably won’t run into problems when dealing with most standard typefaces. But if a given font family offers both regular and condensed fonts for example, you will have to split it into two separate families. See section 8.2 for more details. Finally, NN is fixed to ‘16’. You won’t find the code ‘16’ in the Fontname scheme which only covers 7 and 8-bit encodings since TeX can’t handle 16-bit encodings anyway.

5.3 Running `ttf2tex`

Invoke `ttf2tex` as described in section 4. The *family* argument corresponds to the string `fam` as explained in section 5.1. The names used in conjunction with the `--foundry` and `--font` options must not contain any spaces. By convention, these names will be used to create subdirectories, their sole purpose is to keep your installation clearly arranged.

5.4 Configuring pdfTeX

`ttf2tex` will create the file `fam.map`, where `fam` is the identifier of the font family according to section 5.1. You have to add the name of this map file to the main configuration file for pdfTeX, `pdftex.cfg`, so that it contains a line like `map+fam.map`.

6 Using the fonts

6.1 TrueType fonts

The weight codes employed when renaming the fonts files follow the Fontname scheme while the NFSS uses a different set of codes to determine what is called a ‘series’. You need to know about the NFSS series codes which `ttf2tex` uses when creating font definition files if you want to select weights other than regular and bold. Table 3 lists all supported weights with the corresponding weight and series codes. The Fontname weight codes are what you were using when renaming the

Weight	Fontname	NFSS	
		ttf2tex	fontinst
ultra light, thin, hairline	a	ul*	--
extra light	j	el*	--
light	l	l	l
book	k	k*	m
regular	r	m	m
medium	m	mb	mb
demibold	d	db	db
semibold	s	sb	sb
bold	b	b	b
extra bold	x	eb	eb
heavy	h	hb*	eb
black	c	cb*	eb
ultra bold	u	ub	ub
poster	p	pb*	--
<i>default boldface</i>	d,s,b	bx	--

Table 3: Weights and NFSS series codes with `ttf2tex` extensions (*)

font files while the NFSS series codes are what you will need when selecting a certain weight under LaTeX.

The series codes used by `ttf2tex` correspond to those used by `fontinst`. There is only one minor difference which you will probably not notice in most cases but which might come in handy if you install a very comprehensive font family. By default, `fontinst` supports 11 weights and maps them to eight NFSS series while `ttf2tex` uses a one-to-one mapping scheme for all 14 weights defined in the Fontname scheme. For this purpose six new font series specific to `ttf2tex` are introduced. They are marked with an asterisk in table 3. This difference is normally not visible from a user perspective as `ttf2tex` uses aliases to ensure `fontinst`-like behavior. If, for example, a ‘heavy’ weight is provided, it will be available as both ‘hb’ and ‘eb’ unless an ‘extra bold’ font was provided as well. If a font family comes with multiple weights which would be mapped to ‘eb’ under the `fontinst` scheme, however, these extensions will allow you to conveniently use all of them without modifying any font definition files.

When looking at table 3, you will also notice that one series has a special meaning for `ttf2tex`: the bold extended (`bx`) series. Bold extended is basically a perfectly valid NFSS series and Computer Modern actually provides bold extended fonts. `ttf2tex` however deliberately (ab)uses it to set a default bold face. The fact that `\bfdefault` defaults to `bx` while most font families don’t come with bold extended fonts provides a way to set a font-specific default bold face in the font definition file. For most font families `bx` will probably end up as an alias for `b` in the font definition file, but if one of the more moderate bold weights ‘semibold’ or ‘demibold’ is available, `ttf2tex` will prefer that. Note that this is intended as a fallback mechanism only. If multiple bold weights are available it is a good idea to set `\bfdefault` to the desired value explicitly.

6.2 OpenType fonts

Essentially, there are two types of OpenType fonts: those with PostScript glyph data (CFF fonts, file suffix `.otf`) and those with TrueType glyph data (file suffix `.ttf`). Only the latter variant is supported by `ttf2tex` since it is the only one currently supported by both `ttf2pk` and `pdfTeX`. OpenType fonts with TrueType glyph data (or rather: TrueType fonts with OpenType extensions, because that's what they are in essence) are installed and used like any other TrueType font. Since TeX does not make use of the advanced typesetting features provided by OpenType fonts, the main difference is the wealth of glyphs available in these fonts. To exploit that, you need to run `ttf2tex` with the `--expert` option.

When using this option, `ttf2tex` will create three font families: `fam`, `famx`, and `famj`. `fam` is generated like any other family but uses real small caps instead of 'faked' ones, `famx` adds expert f-ligatures ('ff', 'ffi', 'ffl'), and `famj` adds expert f-ligatures as well as old style figures. Accessing the additional glyphs implies using PostScript glyph names. This is equivalent to using the `--ps-names-only` option, but only where required. This is the case for all fonts of the `famx` and `famj` families as well as all small caps fonts of the `fam` family. Note that this can only work if the font files actually contain valid PostScript glyph names. If you experience problems with missing or faulty glyphs this is most likely not the case – and there is nothing `ttf2tex` can do about that.

7 Tutorial

Let's assume you would like to use Monotype's Times New Roman with LaTeX. You have also prepared everything as explained in section 3. First, create a working directory and copy the font files there. By default, `ttf2tex` will not overwrite files when installing fonts into your TeX tree, but it does assume that the working directory contains nothing but the font files you want to install. Identify the regular, italic, bold, and bold italic versions by viewing the files either with a utility such as `gfontview` or with the bare-bones `ftview` that comes with FreeType (press `q` to close the window, by the way). In the latter case you might want to run `ftdump` to get the PostScript name of the font and to read the copyright notice as found in the header of the font file:

```
gfontview $PWD
ftview -g -r 24 ppem file.ttf
ftdump file.ttf | less
```

The PostScript name will usually tell you about the exact weight while the copyright notice contains the name of the font foundry. Rename the files as explained in section 5.1. You should at least come up with something like this:

```
timesr16.ttf    timesri16.ttf
timesb16.ttf    timesbi16.ttf
```

If you went one step further and decided to use more canonical names derived from the Fontname scheme (in this case we could even use the scheme in a strict manner), the font identifier would be `mns` instead of `times`:

```
mnsr16.ttf      mnsri16.ttf
mnsb16.ttf      mnsbi16.ttf
```

Here `m` denotes Monotype, `ns` Times New Roman, `r`, `ri`, `b`, and `bi` indicate weight and variant and `16` means Unicode encoding. I will use `mns` as the font family name for the remainder of this tutorial. After renaming the font files, call `ttf2tex` with the appropriate font family name:

```
ttf2tex.sh --install --foundry monotype --font timesnew mns
```

You will see a lot of messages on the console. These will probably include warning messages about glyphs not being found, since a few glyphs defined in T1 encoding are missing from the WGL4 glyph set covered by this font. The glyphs which are usually reported as missing in ordinary TrueType fonts include the f-ligatures ‘ff’, ‘ffi’, and ‘fff’. This only means that the missing ligatures will not be typeset as a single glyph but as a sequence of glyphs – just like any other character. Ligatures like these are only found in expert encoded Type 1 PostScript and in OpenType fonts. Ordinary TrueType fonts usually do provide the essential ligatures ‘fi’ and ‘fl’, though. If you want to view a table of all T1 encoded glyphs available to TeX and LaTeX, run plain TeX on `testfont.tex` after installing everything and view the resulting DVI file:

```
echo -e "mnsr8t\n\\table\\bye" | tex testfont.tex
```

The situation is worse for TS1 encoding since it is much more exotic and defines glyphs which are usually not available in text fonts. But you should still get the most common symbols such as currency signs and other frequently used symbols like ‘copyright’ or ‘registered’. You can use the `textcomp` package as usual to get access to these symbols. The following command will create a complete table for TS1 encoding:

```
echo -e "mnsr8c\n\\table\\bye" | tex testfont.tex
```

`ttf2tex` will create a map file for pdfTeX called `mns.map`. You have to add that to the main pdfTeX configuration file, `pdftex.cfg`:

```
% default map file provided by tetex
map pdftex.map
% additional map file created by ttf2tex
map +mns.map
```

That’s it. Now you may create a test document to try your new font out. There are no peculiarities specific to TrueType fonts when selecting them under LaTeX. Just keep in mind that, when using DVI or PostScript as preview format, the first run with the new fonts will take a little longer since `ttf2pk` has to generate all PK fonts required to display the document. Subsequent runs will be much faster because, as with Metafont fonts, PK fonts generated by `ttf2pk` are cached in `$VARTeX FONTS`. To select the fonts simply employ the standard NFSS commands as documented in chapter 2 of the LaTeX font selection guide. This guide ships with TeTeX as `fntguide.dvi` and is also available in PDF format from CTAN.⁷ For example, to select Times New Roman anywhere in your document use a command like:

```
\fontencoding{T1}\fontfamily{mns}\selectfont
```

⁷<http://www.ctan.org/tex-archive/macros/latex/doc/fntguide.pdf>

To use Times New Roman as the default roman typeface for the whole document, redefine `\rmdefault` in the preamble:

```
\renewcommand{\rmdefault}{mns}
```

If your typeface provides more than two weights you can select one of them by using the `\fontseries` command in conjunction with the NFSS series codes listed in table 3. To select the demibold (db) weight for example, use the following command:

```
\fontseries{db}\selectfont
```

Compact font switching commands such as `\textbf` or `\bfseries` will work as usual, but keep in mind that they are using `\bfdefault` as the bold weight. If you want `\textbf` and `\bfseries` to use demibold instead, simply redefine `\bfdefault` accordingly:

```
\renewcommand{\bfdefault}{db}
```

Finally, you might want to write a `.sty` file that sets Times New Roman as the default roman typeface. Here is a sample file, install it as `timesnew.sty`:

```
\NeedsTeXFormat{LaTeX2e}
\ProvidesPackage{timesnew}
\RequirePackage[T1]{fontenc}
\RequirePackage{textcomp}
\renewcommand{\rmdefault}{mns}
\endinput
```

Now all you need to do in order to use your new typeface is to put the command `\usepackage{timesnew}` in your document's preamble.

8 Hints and limitations

8.1 General limitations

There are two limitations when using TrueType fonts with TeX and LaTeX according to `ttf2tex`'s approach, both of which are beyond `ttf2tex`'s control: you can't produce resolution independent PostScript and you can't use slanted fonts in PDF files. The main problem with PostScript is the fact that `dvips` does not support TrueType fonts. When the DVI file is processed, `dvips` implicitly calls `ttf2pk` to render the fonts and embeds them in bitmap format in the PostScript file. Since the rendering quality of `ttf2pk` is excellent, the resulting PostScript will look fine when printed at the corresponding resolution, but keep in mind that it is not portable. When using `dvips`, you face a situation similar to that of Metafont fonts. That doesn't mean that you can't print or view the PostScript file on other machines, but the font quality might be unsatisfactory if the resolutions don't match. If you need truly portable files, use pdfTeX instead.

As mentioned in the introduction, pdfTeX offers native TrueType font support. Compared to Type 1 fonts there is only one minor limitation when producing PDF: it does not support slanting or extending of TrueType fonts. For this reason,

some lines in the map file which `ttf2tex` creates for pdfTeX are commented out. Otherwise pdfTeX would complain whenever it reads its map files. If you try to use ‘faked’ slanted shapes of TrueType fonts with pdfTeX, these fonts will be missing in the resulting PDF file. You can work around this issue by making sure that all PK fonts required to typeset the slanted parts have been generated by `ttf2pk` before you run pdfTeX. But note that this would mean using bitmap fonts since the slanted fonts will end up in bitmap Type 3 format in your PDF file – something you were probably trying to avoid by using vector fonts in the first place. In short, don’t use slanted fonts if you want to produce a PDF file. If you want to make sure that no PK fonts were embedded in a PDF file, simply look at the console output of pdfTeX. Here is a short excerpt of pdfTeX’s output as it is processing a file using the font from the tutorial:

```
</var/spool/texmf/pk/modeless/monotype/timesnew/mnsbo16t.600
pk>{/usr/local/share/texmf/pdftex/T1-WGL4.enc}<mnsbi16.ttf><
mnsb16.ttf></var/spool/texmf/pk/modeless/monotype/timesnew/m
nsro16t.600pk><mnsri16.ttf><mnsr16.ttf>
```

You can tell by this that two types of fonts were embedded in the PDF file: the fonts with a `.ttf` extension are TrueType fonts while the ones with the extension `.600pk` are PK fonts, where 600 is their resolution in dpi. Alternatively, you can use Acrobat Reader’s font information dialog or the `pdffonts` utility that comes with recent versions of xpdf to verify the format of the fonts embedded in the PDF file. All PK fonts will be listed as ‘Type 3’. When printed on a 600 dpi printer, this file will look excellent and you won’t be able to tell the difference between TrueType and PK fonts at all. When viewed with Acrobat Reader however, the slanted fonts will look very poor. You have probably seen this distortion of embedded bitmap fonts which are scaled and anti-aliased before when using Metafont versions of the Computer Modern fonts. Note that Ghostscript (including all frontends based on it) does a much better job when dealing with bitmap Type 3 fonts.

This issue does not affect ‘faked’ small caps as these are simulated by typesetting (tall) caps at a smaller size which does not imply any manipulation of the glyph outlines. It also doesn’t affect `ttf2pk` and all applications which implicitly rely on it when processing DVI files (`dvips`, DVI viewers), since `ttf2pk` is capable of manipulating the glyph outlines of TrueType fonts.

8.2 Limitations specific to `ttf2tex`

`ttf2tex` supports T1 (Cork) and TS1 (Text Companion) encoding only and is thus limited to European languages and to typesetting text. If you want to typeset math, you need to use a different font in math mode. If you don’t change any default settings apart from what is suggested in this manual, this is precisely what is going to happen.

There is no support for multiple widths within one font family. If you have a large font family which comes with regular as well as condensed or narrow fonts you have to treat all widths as separate families. This is in fact a rather academic issue as there are no compact width switching commands anyway. Since the NFSS doesn’t have independent concepts of weight and width, separating the width from the weight by splitting up font families even makes sense in a way. After all, you don’t use a condensed font the way you use boldface.

Variants are limited to upright and italics. Oblique fonts are treated as italics when running `ttf2tex` and have to be renamed accordingly. Like the previous point this is essentially a font naming issue. `ttf2tex` doesn't care if the font file provided for, say, the roman upright shape is in fact a script, an old English, or a titling typeface, but you can't use canonical names for such fonts and you may need to split up the font family.

'Faked' slanted shapes are supported for all typefaces including those which actually provide an oblique variant. Oblique fonts are basically slanted derivatives of the corresponding upright shape. They differ from 'faked' slanted shapes in that they were actually drawn by the font designer and not generated by a machine. The italics of most sans serif and typewriter fonts are in fact oblique shapes. If you install such a typeface with `ttf2tex` and want to use its real oblique shape, simply call the italic version. If you call the slanted version explicitly, you will get a 'faked' slanted shape derived from the upright shape by `ttf2pk`. This is not a limitation but intentional.

8.3 Problems with `ttf2pk`

If `xdvi` and `dvips` cause problems while `pdfTeX` works fine, this usually indicates a problem with `ttf2pk`. As mentioned before, `pdfTeX` features native support for TrueType fonts while `xdvi` and `dvips` depend on `ttf2pk`. Note that this difference is only relevant when creating PDF files. When creating DVI output, it does not matter whether the TeX binary you are using is based on `pdfTeX` or traditional TeX as the DVI file does not contain any fonts. This is a typical error message you might get when running `xdvi` or `dvips`:

```
kpathsea: Running mktexpk --mfmode ljfive --bdpi 600 --mag 1+0/600 --dpi 600 mns
r16t
mktexpk: Running mf \mode:=ljfive; mag:=1+0/600; nonstopmode; input mnsr16t
This is METAFONT, Version 2.7182 (Web2C 7.3.7)
kpathsea: Running mktexmf mnsr16t

! I can't find file 'mnsr16t'.
<*> ...e; mag:=1+0/600; nonstopmode; input mnsr16t

Please type another input file name
! Emergency stop.
<*> ...e; mag:=1+0/600; nonstopmode; input mnsr16t

Transcript written on mfput.log.
grep: mnsr16t.log: No such file or directory
mktexpk: 'mf \mode:=ljfive; mag:=1+0/600; nonstopmode; input mnsr16t' failed to
make mnsr16t.600pk.
```

Whenever `ttf2pk` fails, the script `mktexpk`, which generates PK fonts on demand, falls back to Metafont which will obviously always fail with TrueType fonts. To track down the real cause of the problem, run `ttf2pk` with the `-t` option and the font mentioned in the error message from the command line: `ttf2pk -t mnsr16t`. With a working setup, `ttf2pk` will print the line from its map file that matches the requested font:

```
This is ttf2pk version 1.4
mnsr16t mnsr16.ttf Encoding=T1-WGL4.enc
```

In case of a problem, there are two possible scenarios:

```
This is ttf2pk version 1.4
ttf2pk: ERROR: Cannot open file ttfonts.map.
```

This error message means that `ttf2pk` can't find its map file at all. In this case you should locate `ttfont`s.map on your system and set `$TTF2PKINPUTS` in the global configuration file for `kpathsea`, `texmf.cnf`, as explained in section 3 of this manual.

```
This is ttf2pk version 1.4
ttf2pk: ERROR: Cannot find font mnsr16t in ttfonts.map.
```

This error message means that `ttf2pk` found its map file, but the requested font is not listed. If `ttf2tex` didn't report any errors while installing the fonts, this usually indicates that there are multiple copies of `ttfont`s.map on your system and that `ttf2tex` and `ttf2pk` use different ones. In this case you should verify the path to `ttfont`s.map as given in the header of `ttf2tex.sh`, check the `$TTF2PKINPUTS` variable if set, and make sure that there is only one copy of `ttfont`s.map on your system. After that, fix the map file manually or rerun `ttf2tex` to reinstall the fonts. Please note that `ttf2tex` does not remove old lines from `ttfont`s.map. If there is a problem with `ttfont`s.map, always double-check the file and eliminate duplicate mapping lines.

9 Acknowledgments

`ttf2tex` was inspired by a tutorial⁸ written by Damir Rakityansky which explains how to use TrueType fonts with MikTeX. I would also like to thank Vaggelis Kapoulas, Werner Lemberg, and Bruce D'Arcus for their help, contributions, and feature suggestions.

⁸<http://www.radamir.com/tex>