OpenType Font Definition

Andreas Bühmann

v0.1e – 2005/06/04

Abstract

The package otfontdef provides a simple but powerful key-value-based interface for font definition files. It enables you to vary multiple font attributes according to the font size separately from each other.

1 Motivation

Writing font definition files for professional OpenType fonts is tedious. You have many different aspects like optical size, weight, and letterspacing at your disposal that you might want to vary according to the font size but each one independently of the other. To achieve this with ordinary means involves intersecting size intervals, copying unchanged parts, etc., which leads to a lot of repetition and, in turn, to decreased maintainability.

As an example, imagine that you are about to map the font `T1/MinionPro-OsF/m/it` to external fonts in a font definition file. This is easy if there is exactly one external font:

```
\DeclareFontShape{T1}{MinionPro-OsF}{m}{it}{
      <->     MinionPro-It-osf-t1
}{}
```

But let us assume that you have access to multiple optical sizes of that font and that you would like to use them (for simplicity, let us use only two optical sizes, Caption and Text):

```
      <-8.5>  MinionPro-ItCapt-osf-t1
   <8.5->     MinionPro-It-osf-t1
```

You then decide that it would be nice to increase the weights at small sizes; for instance, use Medium for sizes less than 8 pt and Semibold for sizes less than 6 pt. This implies having to split the first interval twice:

```
      <-6>    MinionPro-SemiboldItCapt-osf-t1
     <6-8>    MinionPro-MediumItCapt-osf-t1
     <8-8.5>  MinionPro-ItCapt-osf-t1
```

If you finally come up with the idea of using a letterspaced version of your font at small sizes (less than 7 pt), you will arrive at this configuration, where each of your design decisions is scattered over up to five lines:

```
\DeclareFontShape{T1}{MinionPro-OsF}{m}{it}{
      <-6>    MinionPro-SemiboldItCapt-osf-l1-t1
     <6-7>    MinionPro-MediumItCapt-osf-l1-t1
     <6-8>    MinionPro-MediumItCapt-osf-t1
     <8-8.5>  MinionPro-ItCapt-osf-t1
   <8.5->     MinionPro-It-osf-t1
}{}
```

This package enables you to specify these decisions directly.

```
\DeclareFontShape{T1}{MinionPro-OsF}{m}{it}{
      <->     otf* MinionPro-It
      <-8.5>  otf* [optical=Capt]
      <-6>    otf* [weight=Semibold]
     <6-8>    otf* [weight=Medium]
      <-7>    otf* [spacing=l1]
}{}
```

The package will then analyze your specifications, combine them, and generate the correct external font names according to some naming scheme.

2 Usage

. . .

3 Implementation

\otf@disable@preamblecmds

In a second we need to temporarily disable all commands that can be used in the preamble only (and that occur in the current version of keyval).

```
\newcommand\otf@disable@preamblecmds{%
  \def\@gobble@optional{%
    \@ifnextchar[\@gobble@optional@{}%]
  }%
  \def\@gobble@optional@[##1]{}%
  \def\NeedsTeXFormat##1{\@gobble@optional}%
  \def\ProvidesPackage##1{\@gobble@optional}%
  \let\DeclareOption\@gobbletwo
  \let\ExecuteOptions\@gobble
  \def\ProcessOptions{\@ifstar\@gobble{}}%
}
```

Try hard to also work when loaded from inside an FD file. We use \space because literal spaces are ignored.

```
\ifx\@nodocument\relax
  \PackageWarningNoLine{otfontdef}{Please\space load\space me\space
  in\space the\space preamble.\MessageBreak
  I'm\space doing\space my\space best\space to\space continue\space anyway}%
  \@ifundefined{define@key}{
    \begingroup
```

keyval uses the space token to define its commands. We make sure that it is the right one. (Space has catcode 9 (ignore) in FD files.)

```
    \catcode32=10
    \otf@disable@preamblecmds
```

I have always waited to find a use for \globaldefs; here it is. It is needed for the definitions in keyval to survive this group and especially the group around the FD file. We must be extremely careful not to execute definitions that we do not want to be global: Hence, we cannot let \ProvidesPackage to \ProvidesFile because it changes catcodes. Let us hope that keyval does not change.

```
    \globaldefs=1
    \input keyval.sty
    \endgroup
  }{}%
\else
  \RequirePackage{keyval}[1999/03/16 v1.13]
\fi
```

\otf@makeglobal We have to make definitions global to allow this package to be used from inside FD files,

```
\newcommand\otf@makeglobal[1]{
  \global\expandafter\let\csname #1\expandafter\endcsname
  \csname #1\endcsname
}
```

but we need not when loaded as a normal package.

```
\ifx\@nodocument\relax\else
  \let\otf@makeglobal\@gobble
\fi
```

\otf@info We only show information on chosen configurations when asked to do so.

```
\newif\ifotf@trace\otf@tracefalse
\newcommand*\otf@info[1]{}
\ifx\@nodocument\relax\else
  \DeclareOption{trace}{\otf@tracetrue}
  \ProcessOptions\relax
  \ifotf@trace
    \def\otf@info#1{\PackageInfo{otfontdef}{#1}}
  \fi
\fi
\otf@makeglobal{otf@info}
```

\ifotf@options

Whether we are looking for options or not when scanning the configuration.

```
\newif\ifotf@options
\otf@optionsfalse
\otf@makeglobal{ifotf@options}
\otf@makeglobal{otf@optionstrue}
\otf@makeglobal{otf@optionsfalse}
```

\otf@keys
\otf@definekey

Only the first occurrence of a key sets the associated value; other occurrences are ignored.

```
\newcommand\otf@keys{}
\newcommand\otf@definekey[1]{%
  \define@key{otf}{#1}{%
    \@ifundefined{otf@@#1}{%
      \@namedef{otf@@#1}{##1}%
    }{}%
  }%
  \expandafter\let\csname otf@@#1\endcsname\relax
  \g@addto@macro\otf@keys{\do{#1}}%
  \otf@makeglobal{otf@@#1}%
  \otf@makeglobal{KV@otf@#1}%
}
```

\otf@default@keys
\otf@definedefault

```
\newcommand\otf@default@keys{}
\newcommand\otf@definedefault[2]{%
  \@namedef{KV@otf@#1@default}{#2}%
  \edef\otf@default@keys{\otf@default@keys,#1}%
  \otf@makeglobal{KV@otf@#1@default}
}
```

Define all font attributes

```
\otf@definekey{family}
\otf@definekey{weight}
\otf@definekey{shape}
\otf@definekey{optical}
\otf@definekey{variant}
\otf@definekey{figures}
\otf@definekey{spacing}
\otf@definekey{encoding}
\otf@definekey{size}
\otf@definekey{scale}
\otf@makeglobal{otf@keys}
```

and how their defaults are determined (if at all).

```
\otf@definedefault{weight}
  {\expandafter\KV@otf@weight\expandafter{\otf@Regular}}
\otf@definedefault{optical}
  {\expandafter\KV@otf@optical\expandafter{\otf@Text}}
\otf@definedefault{variant}
```

```
  {\expandafter\otf@splitname@int\f@family--\@empty}
\otf@definedefault{figures}
  {\expandafter\otf@splitname@int\f@family--\@empty}
\otf@definedefault{encoding}
  {\expandafter\KV@otf@encoding\expandafter{\f@encoding}}
\otf@definedefault{size}
  {\expandafter\KV@otf@size\expandafter{\f@size}}
\otf@definedefault{shape}
  {\expandafter\KV@otf@shape\expandafter{\otf@Regular}}
\otf@makeglobal{otf@default@keys}
```

`\otf@Regular` `\otf@Text` All characters of these strings must have catcode 12 (other). They will be matched against substrings of `\DeclareFontShape` configurations.

```
\newcommand*\otf@Regular{Regular}
\newcommand*\otf@Text{Text}
\newcommand*\otf@Ornaments{Ornaments}
\@onelevel@sanitize\otf@Regular
\@onelevel@sanitize\otf@Text
\@onelevel@sanitize\otf@Ornaments
\otf@makeglobal{otf@Regular}
\otf@makeglobal{otf@Text}
\otf@makeglobal{otf@Ornaments}
```

We register a new size function `otf`, which can then be used in FD files. We procede in two phases when building the configuration for a requested font: In the first phase, we ignore all `otf` entries that give only options (no `\mandatory@arg`); from the first entry that has a mandatory argument we start into the second phase, where we rescan the configuration looking for options.

```
\ifx\@nodocument\relax
  \begingroup
  \def\DeclareSizeFunction#1#2{\endgroup\global\@namedef{s@fct@#1}{#2}}%
  \expandafter
\fi
\DeclareSizeFunction{otf}{%
  \ifotf@options
    \otf@get@options
  \else
    \ifx\mandatory@arg\@empty\else
      \otf@get@external@font
    \fi
  \fi
}
```

`\otf@get@options` Simply process all key-value pairs given in the optional argument.

```
\newcommand\otf@get@options{%
  \@expandtwoargs\setkeys{otf}{\optional@arg}%
}
\otf@makeglobal{otf@get@options}
```

`\otf@splitname@ext`
`\otf@splitname@int`

```
\newcommand\otf@splitname@ext{}
\def\otf@splitname@ext#1-#2-#3\@empty{%
  \def\otf@@family{#1}%
  \def\otf@@shape{#2}%
  \ifx\otf@@shape\@empty
    \let\otf@@shape\relax
  \fi
}
\newcommand\otf@splitname@int{}
\def\otf@splitname@int#1-#2-#3\@empty{%
  \KV@otf@family{#1}%
  \def\@tempa{#2}%
  \ifx\@tempa\otf@Ornaments
    \KV@otf@variant{orn}%
  \else
    \KV@otf@figures{#2}%
  \fi
}
\otf@makeglobal{otf@splitname@ext}
\otf@makeglobal{otf@splitname@int}
```

`\otf@get@external@font` This is the master macro that coordinates the processing. We first determine family and (possibly) shape from the mandatory argument.

```
\newcommand\otf@get@external@font{%
  \expandafter\otf@splitname@ext\mandatory@arg--\@empty
```

We can then rescan the current configuration for further options.

```
  \otf@optionstrue
  \try@size@range
```

If any attribute has not received a value yet, we use its default.

```
  \@expandtwoargs\setkeys{otf}{\otf@default@keys}%
```

After having informed the user about the values we have collected, we build the external font name by applying the appropriate naming scheme.

```
  \begingroup
  \def\do##1{\otf@showoption{##1}\MessageBreak}%
  \otf@info{Using\space configuration\MessageBreak
    \otf@keys for\space font\space\font@name}%
  \endgroup
  \@ifundefined{otf@scheme@@\otf@@family}{\otf@scheme@default}{%
    \@nameuse{otf@scheme@@\otf@@family}%
  }%
  \otf@info{Trying\space to\space load\space external\space font\MessageBreak
    ‘\external@font’}%
}
\otf@makeglobal{otf@get@external@font}
```

`\otf@scheme@default` This is the font naming scheme used in the MinionPro project.

```
\newcommand\otf@head{}
\newcommand\otf@tail{}
\newcommand\otf@scheme@default{%
  \begingroup
  \edef\@tempa{\lowercase{\def\noexpand\otf@tail{%
    \otf@opt\otf@@figures
    \otf@opt\otf@@variant
    \otf@opt\otf@@spacing
    \otf@@encoding
  }}}\@tempa
  \edef\otf@head{%
    \ifx\otf@@weight\otf@Regular\else\otf@@weight\fi
    \otf@format@shape\otf@@shape
    \ifx\otf@@optical\otf@Text\else\otf@@optical\fi}%
  \ifx\otf@head\@empty
    \edef\otf@head{\otf@@family-\otf@Regular}%
  \else
    \edef\otf@head{\otf@@family-\otf@head}%
  \fi
```

Scale the font size if a factor is supplied. Perhaps this should already be done in `\otf@get@external@font` and not in each naming scheme.

```
  \@tempdimb\otf@@size\p@
  \@ifundefined{otf@@scale}{}{%
    \@tempdimb \otf@@scale\@tempdimb
  }%
  \edef\@tempa{\endgroup\def\noexpand\external@font{%
    \otf@head-\otf@tail\space at\space\the\@tempdimb}}%
  \@tempa
}
\otf@makeglobal{otf@scheme@default}
\newcommand\otf@format@shape[1]{%
  \@ifundefined{otf@format@shape@#1}{#1}{\@nameuse{otf@format@shape@#1}}%
}
\newcommand\otf@format@shape@Regular{}%
\newcommand\otf@format@shape@Italic{It}%
\newcommand\otf@format@shape@It{It}%
\otf@makeglobal{otf@format@shape}
\otf@makeglobal{otf@format@shape@Regular}
\otf@makeglobal{otf@format@shape@Italic}
\otf@makeglobal{otf@format@shape@It}
```

`\DeclareFontNamingScheme` We do not make this declaration command global. The who wants to use it should really load the package in preamble.

```
\newcommand*\DeclareFontNamingScheme[1]{%
  \@namedef{otf@scheme@@#1}%
}
```

`\otf@opt` An optional name component, separated from the following ones by a hyphen.

```
\newcommand*\otf@opt[1]{%
```

```
  \ifx\relax#1\@empty\else\if\@empty#1\@empty\else#1-\fi\fi
}
```

\otf@showoption Format an option and its current value for displaying it.

```
\newcommand*\otf@showoption[1]{%
  \@spaces #1\space=\space\@ifundefined{otf@@#1}{<undefined>}{%
    \expandafter\expandafter\expandafter\strip@prefix
    \expandafter\meaning\csname otf@@#1\endcsname}
}
\otf@makeglobal{otf@opt}
\otf@makeglobal{otf@showoption}
```