The fonttable package

Author: Peter Wilson, Herries Press
Maintainer: Will Robertson
will dot robertson at latex-project dot org

v1.6d 2025-03-12

Abstract

The package lets you typeset the characters in a font in tabular and/or running text forms.

Contents

1 Introduction

The fonttable package lets you typeset a font's character set in tabular and/or running text forms.

This manual is typeset according to the conventions of the LaTeX docstrip utility which enables the automatic extraction of the LaTeX macro source files [MG04].

2 The package

The package provides commands to typeset a table of all the glyphs in a given font and to typeset an example of regular text. For font designers it provides commands to typeset a 'test' glyph among sets of glyphs from the font.

`\fnthours` As a convenience, `\fnthours` prints the time of day when the file was processed; it uses the 24 hour clock notation. (The macro `\today` prints the date when the file was processed.)

2.1 Table and texts

`\fonttable` The command
`\fonttable{`⟨*testfont*⟩`}`
typesets a table showing all the glyphs in the ⟨*testfont*⟩, where ⟨*testfont*⟩ is the name of a font file[1] like `cmr10` (for Computer Modern Roman) or `pzdr` (for Zapf Dingbats).

NOTE: The mftinc package [Pak05] for pretty-printing METAFONT code also defines a `\fonttable` macro that is akin to this one. If you want to use both packages together then you can use the following general procedure for when a macro `\macro` is defined in both packA and packB packages.

```
\usepackage{packA}
\let\macroA\macro%   save packA's definition
\let\macro\relax%    undefine \macro
\usepackage{packB}%  now it's packB's definition of \macro
...
\macro %   use the packB defintion
\macroA %  use the packA definition
```

`\xfonttable` The command
`\xfonttable{`⟨*encoding*⟩`}{`⟨*family*⟩`}{`⟨*series*⟩`}{`⟨*shape*⟩`}`
typesets a table showing all the glyphs in the font with encoding ⟨*encoding*⟩ (e.g., T1 or OMS), family ⟨*family*⟩ (e.g., `ppl` for Palatino or `cmbrs` for CM Bright Math (OMS)), font series ⟨*series*⟩ (e.g., `sb` for semibold of `m` for medium), and font shape ⟨*shape*⟩ (e.g., `n` for normal or `sc` for small caps). For example:
`\xfonttable{U}{pzd}{m}{n}`
for Zapf Dingbats.

`\pikfont` The command[2]
`\pikfont{`⟨*encoding*⟩`}{`⟨*family*⟩`}{`⟨*series*⟩`}{`⟨*shape*⟩`}`
selects the font with encoding ⟨*encoding*⟩ (e.g., T1 or OMS), family ⟨*family*⟩ (e.g., `ppl` for Palatino or `cmbrs` for CM Bright Math (OMS)), font series ⟨*series*⟩ (e.g., `sb` for semibold of `m` for medium), and font shape ⟨*shape*⟩ (e.g., `n` for normal or `sc` for small caps). For example:
`\pikfont{T1}{ppl}{m}{sc}`
for Palatino small caps. The size of the font corresponds to the current setting (e.g., `\footnotesize`, `\normalsize`, `\Large`). It can also be changed after being selected by the incantation
`\fontsize{`⟨*size*⟩`}{`⟨*baselineskip*⟩`}\selectfont`
where ⟨*size*⟩ is the normal height and ⟨*baselineskip*⟩ is the distance between text lines; the measurement system is `pts` but just use numbers with no units specified. For example:
`\fontsize{12}{15}\selectfont`
for a 12pt font with 15pts between baselines.

If you are unsure about the meaning of the various arguments of `\xfonttable` and `\pikfont` see *The Companion* [MG04, Chapter 7] or the *LaTeX2e font selection* manual (`fntguide.tex`; try `texdoc fntguide`).

`\fontrange` The package attempts to populate the table with a maximum of 256 glyphs,

[1] More precisely, the name of a `.tfm` file.

[2] The name was chosen in an attempt to avoid clashes with other macros that might perform similar functions.

numbered from 0 to 255. The `\fontrange{`⟨*low*⟩`}{`⟨*high*⟩`}` declaration changes this by reducing the range so that it extends from ⟨*low*⟩ to ⟨*high*⟩, where ⟨*low*⟩ should be at least 0 and ⟨*high*⟩ at most 256, and ⟨*low*⟩ less than ⟨*high*⟩.

The table is composed of blocks of sixteen characters. If necessary the value of ⟨*low*⟩ is adjusted lower and ⟨*high*⟩ is adjusted higher to match this block structure. For example, if you wanted a table of the lower 128 characters then `\fontrange{0}{127}` would do the job, while the upper half of a 256 character font could be tabulated via `\fontrange{128}{255}`.

`\decimals` `\nodecimals` Normally each cell in the table includes the decimal number of the position in the (256) character set. `\nodecimals` turns off this numbering and `\decimals` turns it on. The default is `\decimals`.

`\hexoct` `\nohexoct` Normally the columns and rows in the table are numbered using hexadecimal and octal numbers. These can be turned off by `\nohexoct` and turned on again with `\hexoct`, which is the default.

`\ftablewidth` The font table's width is the length `\ftablewidth`, which by default is set to the normal textwidth (or more exactly, to `\hsize`). The table itself is left aligned. However, if `\nohexoct` is in effect the width of the table is its natural width.

`\fntcolwidth` When `\nohexoct` is in effect the minimum width of a table column is `\fntcolwidth`. This is initially declared as
`\setwidth{\fntcolwidth}{0.08\ftablewidth}`

`\fonttext` The command `\fonttext{`⟨*testfont*⟩`}` typesets an example text using the ⟨*testfont*⟩ (e.g. `cmr10`).

`\simpletext` `\fulltext` The example text can be just a paragraph and a line of capitals, or include more complex accented words as well. Following the declaration `\fulltext` the complex words are included as well as the example paragraph. The default is `\simpletext` for just the paragraph.

`\regulartext` The command `\regulartext{`⟨*fontspec*⟩`}` typesets the example text using ⟨*fontspec*⟩, for example `\rmfamily\itshape` or `\pikfont{T1}{pnc}{m}{it}`.

`\fonttexts` `\regulartexts` The macro `\fonttexts{`⟨*testfont*⟩`}{`⟨*text*⟩`}` typesets ⟨*text*⟩ using the ⟨*testfont*⟩ (e.g., `cmr10`). Similarly the macro `\regulartexts{`⟨*fontspec*⟩`}{`⟨*text*⟩`}` typesets ⟨*text*⟩ using ⟨*fontspec*⟩ (e.g., `\rmfamily\itshape` or `\pikfont{T1}{ppl}{m}{it}`).

`\germanparatext` `\latinparatext` `\germanparatext` expands to a German language paragraph, borrowed from the blindtext package [Lik05]. `\latinparatext` expands to one version of a paragraph of the traditional *lorem ipsum* dummy Latin text. Either, or both, of these could be used as the ⟨*text*⟩ argument to `\fonttexts` or `\regulartexts`.
NOTE: These were originally called `\germantext` and `\latintext` but on 2009/05/14 I was told that the babel package defines `\latintext`, which causes unexpected results if it is used in the same document as this package. To try and be on the safe side I renamed `\germantext` as well as `\latintext`.

`\aztext` `\AZtext` `\digitstext` `\punctext` `\aztext` expands to the lowercase Latin alphabet a to z, and `\AZtext` is the corresponding command for the uppercase A to Z. The macros `\digitstext` and `\punctext` expand respectively to the digits 0 to 9, and to the typical punctuation marks. In all cases there is a space between each character.

2.2 Testing a glyph

The macros here are a reimplementation of Donald Knuth's `testfont.tex`, which is available from CTAN.

In the following, the value of a glyph argument can be specified as its location in the font (i.e., as a decimal number). With a few exceptions, if the glyph is

within the visible ASCII range (33–126) it may instead be specified by the ASCII character prefixed with a single open quote mark[3] (‘). The exceptions are nos: 37 (%), 92 (\) 123 ({) and 125 (}) (but there may be others). In any case, the glyph representing the character p can be specified either as ‘p or as 112.

The glyphs are taken from the current font. If the font does not have Latin alphabet glyphs in the ASCII locations then in the descriptions below phrases like ‘lowercase alphabet’ or ‘uppercase alphabet’ or ‘digits’, should be taken to mean (the glyphs in) those locations.

\glyphmixture `\glyphmixture{⟨T⟩}{⟨S⟩}{⟨E⟩}` typesets the ⟨T⟩ (test) glyph between the glyphs in the range from ⟨S⟩ (start) to ⟨E⟩ (end). For example
`\glyphmixture{‘e}{‘f}{‘g}` will produce
efeeffeeefffef
egeeggeeegggeg

\glyphalternation `\glyphalternation{⟨T⟩}{⟨S⟩}{⟨E⟩}` typesets the ⟨T⟩ glyph alternately between each glyph in the range from ⟨S⟩ to ⟨E⟩. For example
`\glyphalternation{‘e}{‘f}{‘g}` will produce
efefefefefefefe
egegegegegegege

\glyphseries `\glyphseries{⟨T⟩}{⟨S⟩}{⟨E⟩}` typesets the ⟨T⟩ glyph between the glyphs in the range from ⟨S⟩ to ⟨E⟩. For example
`\glyphseries{‘e}{‘f}{‘h}` will produce
efegehe

\glyphalphabet `\glyphalphabet{⟨T⟩}` typesets the ⟨T⟩ glyph between each letter of the lowercase Latin alphabet plus a few others. `\GLYPHALPHABET{⟨T⟩}` does the same but using the uppercase Latin alphabet. For example, the output of
`\glyphalphabet}{‘3}` is like
3a3b3c3d3e3f3g...3z3Ø3˜3!3"3

\glyphlowers `\glyphlowers` takes each character of the lowercase alphabet in turn as a test glyph and sets it interspersed among the other lowercase characters. `\glyphuppers` and `\glyphdigits` are similar except that they use the uppercase alphabet and the ten digits instead. For example, `\glyphdigits` produces output like
00010203040506070809 0
10111213141516171819 1
20212223242526272829 2
...
90919293949596979899 9

\glyphpunct `\glyphpunct` sets a collection of words with an assortment of punctuation marks.

3 The code

1 ⟨*pack⟩

3.1 Table and texts

Most of the code below is an edited version of code used in `nfssfont.tex` for displaying aspects of the set of glyphs in a font.

[3] Sometimes called a ‘backquote’.

\sevenrm A small fixed size roman font.

```
\providecommand*{\sevenrm}{\fontsize{7}{9pt}\rmfamily}
```

\f@tm Counts and a dimen.
\f@tn
\f@tp
\f@tdim

```
\newcount\f@tm \newcount\f@tn \newcount\f@tp \newdimen\f@tdim

```

\fonttable \fonttable{⟨*font*⟩} typesets a table of all the glyphs in the ⟨*font*⟩ (e.g., auncl10).

```
\newcommand*{\fonttable}[1]{%
  \def\f@tfontname{#1}%
  \bgroup
  \f@tstartfont
  \ftable
  \egroup}

```

\pikfont \pikfont{⟨*encoding*⟩}{⟨*family*⟩}{⟨*series*⟩}{⟨*shape*⟩} selects the font with ⟨*encoding*⟩, ⟨*family*⟩, ⟨*series*⟩ and ⟨*shape*⟩.

```
\DeclareRobustCommand{\pikfont}[4]{%
  \fontencoding{#1}\fontfamily{#2}\fontseries{#3}\fontshape{#4}\selectfont}

```

\xfonttable \xfonttable{⟨*encoding*⟩}{⟨*family*⟩}{⟨*series*⟩}{⟨*shape*⟩} typesets a table of all the glyphs in the font with ⟨*encoding*⟩, ⟨*family*⟩, ⟨*series*⟩ and ⟨*shape*⟩ (e.g., `\xfonttable{T1}{pnc}{m}{it}` for New Century Schoolbook italic). The original code for the macro was supplied by Enrico Gregorio.

```
\newcommand*{\xfonttable}[4]{%
  \begingroup
    \pikfont{#1}{#2}{#3}{#4}%
    \edef\f@tfontname{\fontname\font}%
```

New: strip any size information from the fontname (which could be, e.g., either '`cmr10`' or '`cmr10␣at␣10pt`'.) This wasn't necessary before because we didn't explicitly choose the font size; it was inferred automatically.

```
    \edef\@tempa{ \string a\string t}%
    \edef\@tempb{\noexpand\in@{\@tempa}{\f@tfontname}}%
    \@tempb
    \ifin@
      \edef\f@tfontname{\expandafter\f@tstripsize\f@tfontname}%
    \fi
```

End new code, and finish as before:

```
    \normalfont
    \f@tstartfont
    \ftable
  \endgroup
}
```

\f@tstripsize Needed above.

```
\edef\@tempa{%
  \def\noexpand\f@tstripsize
    ##1 \string a\string t##2\string p\string t{##1}%
}
\@tempa
```

\f@tstartfont Sets up for a font table.

```
\newcommand*{\f@tstartfont}{%
```

New: scale the font by 0.01% to (attempt to) avoid TeX's font optimisation. This becomes a problem in Spanish babel, say, when \textfont\fam changes when cmr10 has been loaded under a different name, here. (And the \textfont can no longer be parsed correctly. See: http://latex-alive.tumblr.com/post/3229118083/texs-font-loading-optimisation)

```
  \@tempdima=\f@size pt
  \font\f@ttestfont=\f@tfontname\space at 0.9999\@tempdima\relax
```

Continue as before:

```
  \f@ttestfont \f@tsetbaselineskip
  \ifdim\fontdimen6\f@ttestfont<10pt\relax
    \rightskip=0pt plus 20pt\relax
  \else
    \rightskip=0pt plus 2em\relax
  \fi
  \spaceskip=\fontdimen2\f@ttestfont % space between words (\raggedright)
  \xspaceskip=\fontdimen2\f@ttestfont
  \advance\xspaceskip by\fontdimen7\f@ttestfont
}
```

\f@tsetbaselineskip

```
\newcommand*{\f@tsetbaselineskip}{\setbox0=\hbox{\f@tn=0
  \loop\char\f@tn \ifnum \f@tn<255 \advance\f@tn 1 \repeat}
  \baselineskip=6pt \advance\baselineskip\ht0 \advance\baselineskip\dp0 }

```

\f@toct \f@toct{⟨*onum*⟩} typesets the octal constant ⟨*onum*⟩.

```
\newcommand*{\f@toct}[1]{\hbox{\rmfamily\'{}\kern-.2em\itshape
          #1\/\kern.05em}} % octal constant
```

\f@thex \f@thex{⟨*hnum*⟩} typesets the hexadecimal constant ⟨*hnum*⟩.

```
\newcommand*{\f@thex}[1]{\hbox{\rmfamily\H{}\ttfamily#1}} % hexadecimal constant
```

\f@tsetdigs \f@tsetdigs

```
\def\f@tsetdigs#1"#2{\gdef\h{#2}% \h=hex prefix; \0\1=corresponding octal
 \f@tm=\f@tn \divide\f@tm by 64 \xdef\0{\the\f@tm}%
 \multiply\f@tm by-64 \advance\f@tm by\f@tn \divide\f@tm by 8 \xdef\1{\the\f@tm}}
```

\f@ttestrow \f@ttestrow checks if there are any characters in the next block of 16 slots.

```
\newcommand*{\f@ttestrow}{\setbox0=\hbox{\penalty 1\def\\{\char"\h}%
 \\0\\1\\2\\3\\4\\5\\6\\7\\8\\9\\A\\B\\C\\D\\E\\F%
 \global\f@tp=\lastpenalty}} % \f@tp=1 if none of the characters exist

```

\ifhexoct Flag for (not) setting hex and octal numbers.
\hexoct
\nohexoct

```
\newif\ifhexoct
\newcommand*{\hexoct}{\hexocttrue}
\newcommand*{\nohexoct}{\hexoctfalse}
\hexoct

```

\f@toddlinenum \f@toddline

```
\newcommand*{\f@toddline}{\cr
  \noalign{\nointerlineskip}
  \multispan{19}\hrulefill&
  \setbox0=\hbox{\lower 2.3pt\hbox{\f@thex{\h x}}}\smash{\box0}
  \cr
  \noalign{\nointerlineskip}}

```

\iff@tskipping
\f@tskippingtrue
\f@tskippingfalse

```
\newif\iff@tskipping

```

\fontrange `\fontrange{`⟨*low*⟩`}{`⟨*high*⟩`}` sets the character range to be output.

```
\newcommand*{\fontrange}[2]{%
  \ifnum#1<#2\relax
```

Set `\f@tlow` to the nearest multiple of 16 that is at or below ⟨*low*⟩, but first make sure that it will be at least 0.

```
  \ifnum#1<\z@
    \f@tm=\z@
  \else
    \f@tm=#1
    \divide \f@tm \sixt@@n
    \multiply \f@tm \sixt@@n
  \fi
  \edef\f@tlow{\the\f@tm}
```

Set `\f@thigh` to the nearest multiple of 16 at or above ⟨*high*⟩, finally making sure that its maximum is 256.

```
  \f@tm=#2
  \divide \f@tm \sixt@@n
  \advance \f@tm \@ne
  \multiply \f@tm \sixt@@n
  \ifnum \f@tm > \@cclvi  \f@tm=\@cclvi \fi
  \edef\f@thigh{\the\f@tm}
\else
  \PackageError{fonttable}{%
     Improper values for fontrange. Default values substituted}{\@ehc}
  \def\f@tlow{0} \def\f@thigh{256}
  \fi}
\fontrange{0}{256}

```

\f@tloopforsixteen `\f@tloopforsixteen` sets up a block of sixteen character slots.

```
\newcommand*{\f@tloopforsixteen}{%
  \ifnum\f@tn<\f@tlow \global\f@tn=\f@tlow\fi
  \loop\f@tskippingfalse
  \ifnum\f@tn<\f@thigh \f@tm=\f@tn \divide\f@tm \sixt@@n \chardef\next=\f@tm
  \expandafter\f@tsetdigs\meaning\next \f@ttestrow
  \ifnum\f@tp=\@ne \f@tskippingtrue \fi\fi
  \iff@tskipping \global\advance\f@tn \sixt@@n \repeat}

```

\f@tevenline
\f@tevenlinenonum

\f@tevenline gets next non-empty set of a block of 16 characters. It either calls \f@tmorechart to print them, or \f@tendchart to finish off the table if all 256 potential characters have been processed.

\f@tevenlinenonum does something similar when no external numbers are printed.

```
\newcommand*{\f@tevenline}{%
  \f@tloopforsixteen
  \ifnum\f@tn=\f@thigh \let\next=\f@tendchart\else\let\next=\f@tmorechart\fi
  \next}
\newcommand*{\f@tevenlinenonum}{%
  \f@tloopforsixteen
  \ifnum\f@tn=\f@thigh\relax
    \\\hline
  \else
    \\\hline
    \f@tmorechartnonum
  \fi}
```

\f@tmorechart
\f@tmorechartnonum

\f@tmorechart sets two lines of the table, and \f@tmorechartnonum does the same when there are no external numbers.

```
\newcommand*{\f@tmorechart}{\cr\noalign{\hrule\penalty5000}
 \f@tchartline \f@toddline \f@tm=\1 \advance\f@tm 1 \xdef\1{\the\f@tm}
 \f@tchartline \f@tevenline}
\newcommand*{\f@tmorechartnonum}{%
  \f@tsimpleline \\ \hline
  \f@tsimpleline \f@tevenlinenonum}
```

\f@tchartline
\f@tsimpleline

\f@tchartline does a line of the table, including external numbers, and \f@tsimpleline does an unnumbered line.

```
\newcommand*{\f@tchartline}{%
  &\f@toct{\0\1x}&&\f@tpsg{}&&\f@tpsg{}&&\f@tpsg{}&&\f@tpsg{}&&\f@tpsg{}&&\f@tpsg{}&&\f
\newcommand*{\f@tsimpleline}{%
  \f@tpsg{}\f@tchartstrut& \f@tpsg{} & \f@tpsg{} & \f@tpsg{} & \f@tpsg{} & \f@tpsg{} &
```

\f@tchartstrut
\ftablewidth
\fntcolwidth

\f@tchartstrut is a strut used in each table line. \ftablewidth is width of an externally numbered table. \fntcolwidth is the minimum width of a column in an unnumbered table.

```
\newcommand*{\f@tchartstrut}{\lower4.5pt\vbox to14pt{}}
\newdimen\ftablewidth
  \ftablewidth=\hsize
\newdimen\fntcolwidth
  \setlength{\fntcolwidth}{0.08\ftablewidth}
```

\f@tcol
\f@tstartchartnonum

\f@tstartchartnonum is a table line of spaces, with no verticals.

```
\newcommand*{\f@tcol}{%
  \multicolumn{1}{c}{\hspace*{\fntcolwidth}}}
\newcommand*{\f@tstartchartnonum}{%
  \f@tcol &\f@tcol &\f@tcol &\f@tcol &\f@tcol &\f@tcol &\f@tcol &\f@tcol}
```

\ftable \f@tftablenum \f@tftablenonum

\ftable sets a complete character table. The actual code is in either \f@tftablenum or \f@tftablenonum for externally numbered or plain tables, respectively.

```
\newcommand*{\f@tftablenum}{$$\global\f@tn=\z@
  \halign to\ftablewidth\bgroup
    \f@tchartstrut##\tabskip0pt plus10pt&
    &\hfil##\hfil&\vrule##\cr
    \lower6.5pt\null
    &&&\f@toct0&&\f@toct1&&\f@toct2&&\f@toct3&&\f@toct4&&\f@toct5&&\f@toct6&&\f@toct7&%
    \f@tevenline}
\newcommand*{\f@tftablenonum}{%
  \global\f@tn=\z@
  \begin{tabular}{|c|c|c|c|c|c|c|c|}
    \f@tstartchartnonum
    \f@tevenlinenonum
  \end{tabular}}
\newcommand*{\ftable}{\ifhexoct\f@tftablenum\else\f@tftablenonum\fi}

```

\f@tendchart

\f@tendchart sets the last line of an externally numbered table with the relevant hex digits.

```
\newcommand*{\f@tendchart}{\cr\noalign{\hrule}
  \raise11.5pt\null&&&\f@thex 8&&\f@thex 9&&\f@thex A&&\f@thex B&
  &\f@thex C&&\f@thex D&&\f@thex E&&\f@thex F&\cr
  \egroup$$\par}

```

\f@tpsg \f@placechar \f@placedecimal

\f@tpsg typesets a single glyph, possibly with its decimal slot number. \f@placechar is the function to typeset the glyph with its number that is internally defined as \f@placedecimal if decimals are to be shown.

```
\newcommand*{\f@tpsg}{%
  \setbox\z@=\hbox{\f@placechar{\char\f@tn}{\the\f@tn}}%
  \ifdim\ht\z@>7.5pt\relax
    \f@treposition
  \else
    \ifdim\dp\z@>2.5pt\relax
      \f@treposition
    \fi
  \fi
  \box\z@
  \global\advance\f@tn\@ne
}
```

Change this definition to adjust the typesetting of the decimal numbers:

```
\newcommand*\f@placedecimal[2]{#1\ {\tiny #2}}
```

\decimals \nodecimals

Following \decimals, which is the default, decimal numbers are printed in the table. Following \nodecimals they are not printed.

```
\newcommand*{\nodecimals}{%
  \renewcommand*\f@placechar{\@firstoftwo}%
}

\newcommand{\decimals}{%
  \renewcommand*\f@placechar{\f@placedecimal}%
}
```

```
\newcommand*\f@placechar{}
\decimals
```

\f@treposition

```
\f@treposition
\newcommand*{\f@treposition}{\setbox0=\vbox{\kern2pt\box0}\f@tdim=\dp0
  \advance\f@tdim 2pt \dp0=\f@tdim}
```

\fonttext `\fonttext{⟨font⟩}` typesets `\knutext` using ⟨*font*⟩ (e.g. auncl10).

```
\def\fonttext#1{%
  \def\f@tfontname{#1}%
  \bgroup
  \f@tstartfont
  \knutext
  \egroup}
```

\regulartext `\regulartext{⟨fontspec⟩}` typesets `\knutext` using ⟨*fontspec*⟩ (e.g., `\aunclfamily`).

```
\def\regulartext#1{%
  \bgroup
  #1
  \knutext
  \egroup}
```

\knutext Deathless prose from Knuth for testing a font. It includes `\moreknutext`, `\capknutext`, and `\knunames`.

```
\def\knutext{{
On November 14, 1885, Senator \& Mrs.~Leland Stanford called together
at their San Francisco mansion the 24~prominent men who had been
chosen as the first trustees of The Leland Stanford Junior University.
They handed to the board the Founding Grant of the University, which
they had executed three days before. This document---with various
amendments, legislative acts, and court decrees---remains as the
University's charter.  In bold, sweeping language it stipulates that
the objectives of the University are ``to qualify students for
personal success and direct usefulness in life; and to promote the
publick welfare by exercising an influence in behalf of humanity and
civilization, teaching the blessings of liberty regulated by law, and
inculcating love and reverence for the great principles of government
as derived from the inalienable rights of man to life, liberty, and
the pursuit of happiness.''

\moreknutext

\capknutext

\knunames
\par}}
```

\@moreknutext Some more text with a variety of ligatures and accents.

```
\def\@moreknutext{?`But aren't Kafka's Schlo{\ss} and {\AE}sop's
```

```
{\OE}uvres often na{\"\i}ve vis-\‘a-vis the d{\ae}monic ph{\oe}nix’s
official r\^ole in fluffy souffl\’es? }

```

\@capknutext Text using only capital letters and some punctuation.
\capknutext

```
\newcommand{\@capknutext}{%
(!‘THE DAZED BROWN FOX QUICKLY GAVE 12345--67890 JUMPS!)}
\let\capknutext\@capknutext

```

\@knunames Lots of accents masquerading in personal names.

```
\def\@knunames{ {\AA}ngel\aa\ Beatrice Claire
  Diana \’Erica Fran\c{c}oise Ginette H\’el\‘ene Iris
  Jackie K\=aren {\L}au\.ra Mar{\’\i}a N\H{a}ta{\l}{\u\i}e {\O}ctave
  Pauline Qu\^eneau Roxanne Sabine T\~a{\’\j}a Ur\v{s}ula
  Vivian Wendy Xanthippe Yv{\o}nne Z\"azilie\par}

```

\guillemotleft Just in case the French quotes are not defined, as they are called for in the subse-
\guillemotright quent \germantext.
\flqq
\frqq

```
\DeclareTextSymbol{\guillemotleft}{OT1}{‘\’}
\DeclareTextSymbol{\guillemotright}{OT1}{‘\‘}
\providecommand{\flqq}{\guillemotleft}
\providecommand{\frqq}{\guillemotright}

```

\germantext Text from the Blindtext package.
\germanparatext

```
\providecommand*{\germantext}{%
  \PackageWarning{fonttable}{\protect\germantext\space is deprecated,
                  \MessageBreak use \protect\germanparatext\space instead}}
\newcommand*{\germanparatext}{%
Dies hier ist ein Blindtext zum Testen von Textausgaben. Wer
diesen Text liest, ist selbst schuld. Der Text gibt lediglich den
Grauwert der Schrift an. Ist das wirklich so? Ist es
gleich\-g\"ul\-tig ob ich schreibe: \frqq Dies ist ein
Blindtext\flqq\ oder \frqq Huardest gefburn\flqq? Kjift --
mitnichten! Ein Blindtext bietet mir wichtige Informationen. An
ihm messe ich die Lesbarkeit einer Schrift, ihre Anmutung, wie
harmonisch die Figuren zueinander stehen und pr\"u\-fe, wie breit
oder schmal sie l\"auft. Ein Blindtext sollte m\"og\-lichst viele
verschiedene Buchstaben enthalten und in der Originalsprache
gesetzt sein. Er mu\ss\ keinen Sinn ergeben, sollte aber lesbar
sein. Fremdsprachige Texte wie \frqq Lorem ipsum\flqq\ dienen
nicht dem eigentlichen Zweck, da sie eine
falsche Anmutung vermitteln.\par}

```

\latintext The traditional printers’ text.
\latinparatext

```
\providecommand*{\latintext}{%
  \PackageWarning{fonttable}{\protect\latintext\space may be overridden by the
   babel package \MessageBreak use
                  \protect\latinparatext\space instead}}
\newcommand*{\latinparatext}{%
```

```
265 Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Etiam
266 lobortis facilisis sem. Nullam nec mi et neque pharetra
267 sollicitudin. Praesent imperdiet mi nec ante. Donec ullamcorper,
268 felis non sodales commodo, lectus velit ultrices augue, a
269 dignissim nibh lectus placerat pede. Vivamus nunc nunc, molestie
270 ut, ultricies vel, semper in, velit. Ut porttitor. Praesent in
271 sapien. Lorem ipsum dolor sit amet, consectetuer adipiscing elit.
272 Duis fringilla tristique neque. Sed interdum libero ut metus.
273 Pellentesque placerat. Nam rutrum augue a leo. Morbi sed elit sit
274 amet ante lobortis sollicitudin. Praesent blandit blandit mauris.
275 Praesent lectus tellus, aliquet aliquam, luctus a, egestas a,
276 turpis. Mauris lacinia lorem sit amet ipsum. Nunc quis urna dictum
277 turpis accumsan semper.\par}
278
```

`\simpletext` `\fulltext` `\moreknutext` `\knunames`

`\simpletext` kills off `\moreknutext` and `\knunames`. `\fulltext` restores `\moreknutext` and `\knunames`. Make `\fulltext` the default.

```
279 \newcommand*{\simpletext}{\let\moreknutext\relax \let\knunames\relax}
280 \newcommand*{\fulltext}{\let\moreknutext\@moreknutext \let\knunames\@knunames}
281 \fulltext
282
```

`fonttexts`

`\fonttexts{`⟨*font*⟩`}{`⟨*text*⟩`}` typesets ⟨*text*⟩ using ⟨*font*⟩ (e.g. auncl10).

```
283 \def\fonttexts#1#2{%
284   \def\f@tfontname{#1}%
285   \bgroup
286   \f@tstartfont
287   #2
288   \egroup}
289
```

`\regulartexts`

`\regulartext{`⟨*fontspec*⟩`}{`⟨*text*⟩`}` typesets ⟨*text*⟩ using ⟨*fontspec*⟩ (e.g., `\aunclfamily`).

```
290 \def\regulartexts#1#2{%
291   \bgroup
292   #1 #2
293   \egroup}
294
```

`\aztext` `\AZtext` `\digitstext` `\punctext`

The various characters used for Latin texts.

```
295 \newcommand*{\aztext}{a b c d e f g h i j k l m n o p q r s t u v w x y z}
296 \newcommand*{\AZtext}{A B C D E F G H I J K L M N O P Q R S T U V W X Y Z}
297 \newcommand*{\digitstext}{0 1 2 3 4 5 6 7 8 9}
298 \newcommand*{\punctext}{‘ ! @ \$ \& * ( ) \_ - + =  [ ] < > \{ \} : ; ’ , . ? /}
299
```

3.2 Testing a glyph

This is a reimplementation of Donald Knuth's `testfont.tex` which is available from CTAN and there is also a commented version in Appendix H of *The METAFONT Book*.

`\fnthours` `\f@ttwodigits`

The time of day on a 24 hour clock.

```
300 %%%%%%%% using \@tempcnta for Knuth's \m and \@tempcntb for his \n
```

```
\newcommand*{\fnthours}{\@tempcntb=\time \divide\@tempcntb 60
  \@tempcnta=-\@tempcntb \multiply\@tempcnta 60 \advance\@tempcnta \time
  \f@ttwodigits\@tempcntb:\f@ttwodigits\@tempcnta}
\newcommand*{\f@ttwodigits}[1]{\ifnum #1<10 0\fi \number#1}
```

`\f@tgettsechars` `\f@ttchar` `\f@tschar` `\f@techar`

`\f@tgettsechars{`⟨*T*⟩`}{`⟨*S*⟩`}{`⟨*E*⟩`}` gets three characters and `\chardef`s `\f@ttchar` to ⟨*T*⟩ (the test character), `\f@tschar` to ⟨*S*⟩ (start character) and `\f@techar` to ⟨*E*⟩ (the end character).

```
\newcommand*{\f@tgettsechars}[3]{%
  \chardef\f@ttchar=#1 \chardef\f@tschar=#2 \chardef\f@techar=#3}
```

`\glyphmixture` `\f@tmixpattern` `\f@tdomix`

`\glyphmixture{`⟨*T*⟩`}{`⟨*S*⟩`}{`⟨*E*⟩`}` sets a mix of ⟨*T*⟩ within the glyph range from ⟨*S*⟩ to ⟨*E*⟩ according to the pattern `\f@tmixpattern`. The work is done by `\f@tdomix`.

```
\newcommand*{\glyphmixture}[3]{\f@tgettsechars{#1}{#2}{#3}%
                                  \f@tdomix\f@tmixpattern}
\newcommand*{\f@tmixpattern}{\0\1\0\0\1\1\0\0\0\1\1\1\0\1}
\newcommand*{\f@tdomix}[1]{\par\chardef\0=\f@ttchar \@tempcntb=\f@tschar
   \loop \chardef\1=\@tempcntb #1\endgraf
   \ifnum \@tempcntb<\f@techar \advance\@tempcntb \@ne \repeat}
```

`\glyphalternation` `\f@taltpattern`

These are similar to `\glyphmixture` and `\f@tmixpattern` except that the glyphs are alternated.

```
\newcommand*{\glyphalternation}[3]{\f@tgettsechars{#1}{#2}{#3}%
                                     \f@tdomix\f@taltpattern}
\newcommand*{\f@taltpattern}{\0\1\0\1\0\1\0\1\0\1\0\1\0\1\0\1\0}
```

`\f@tdisc`

For breaking long lines so that the test character will be at the end of one line and repeated at the start of the next one.

```
\newcommand*{\f@tdisc}{\discretionary{\f@ttchar}{\f@ttchar}{\f@ttchar}}
```

`\glyphseries` `\f@tdoseries`

`\glyphseries{`⟨*T*⟩`}{`⟨*S*⟩`}{`⟨*E*⟩`}` puts the test character ⟨*T*⟩ between all the others in the range ⟨*S*⟩ to ⟨*E*⟩. The work is done by `\f@tdoseries`.

```
\newcommand*{\glyphseries}[3]{\f@tgettsechars{#1}{#2}{#3}%
  \f@tdisc\f@tdoseries\f@tschar\f@techar\par}
\newcommand*{\f@tdoseries}[2]{\@tempcntb=#1\relax
  \loop\char\@tempcntb\f@tdisc
    \ifnum\@tempcntb<#2\advance\@tempcntb \@ne \repeat}
```

`\glyphalphabet` `\GLYPHALPHABET` `\f@tcomplower` `\f@tcompupper`

`\glyphalphabet{`⟨*T*⟩`}` inserts the test glyph ⟨*T*⟩ between the lowercase alphabetic characters. Similarly `\GLYPHALPHABET{`⟨*T*⟩`}` does the same with the uppercase characters. The work is done by, respectively, `\f@tcomplower` and `\f@tcompupper`.

```
\newcommand*{\glyphalphabet}{\f@tcomplower}
\newcommand*{\GLYPHALPHABET}{\f@tcompupper}
\newcommand*{\f@tcomplower}[1]{\chardef\f@ttchar=#1
```

```
  \f@tdisc\f@tdoseries{‘a}{‘z}\f@tdoseries{31}{34}\par}
\newcommand*{\f@tcompupper}[1]{\chardef\f@ttchar=#1
  \f@tdisc\f@tdoseries{‘A}{‘Z}\f@tdoseries{35}{37}\par}
```

`\glyphlowers` `\glyphuppers` `\glyphdigits` `\f@tclc` `\f@tcuc` `\f@tdgs` `\f@tdocomprehensive`

These macros generate an extended mix of characters of a particular kind. The work is done by `\f@tdocomprensive` with `\f@tclc`, `\f@tcuc`, and `\f@tdgs` setting up the glyph sets.

```
\newcommand*{\glyphlowers}{\f@tdocomprehensive\f@tclc{‘a}{‘z}{31}{34}}
\newcommand*{\glyphuppers}{\f@tdocomprehensive\f@tcuc{‘A}{‘Z}{35}{37}}
\newcommand*{\glyphdigits}{\f@tdocomprehensive\f@tdgs{‘0}{‘4}{‘5}{‘9}}
\newcommand*{\f@tdocomprehensive}[5]{\par\chardef\f@ttchar=#2
  \loop{#1} \ifnum\f@ttchar<#3\@tempcnta=\f@ttchar\advance\@tempcnta \@ne
  \chardef\f@ttchar=\@tempcnta \repeat
  \chardef\f@ttchar=#4
  \loop{#1} \ifnum\f@ttchar<#5\@tempcnta=\f@ttchar\advance\@tempcnta \@ne
  \chardef\f@ttchar=\@tempcnta \repeat}
\newcommand*{\f@tclc}{\f@tdisc\f@tdoseries{‘a}{‘z}\f@tdoseries{31}{34}\par}
\newcommand*{\f@tcuc}{\f@tdisc\f@tdoseries{‘A}{‘Z}\f@tdoseries{35}{37}\par}
\newcommand*{\f@tdgs}{\f@tdisc\f@tdoseries{‘0}{‘9}\par}
```

`\glyphpunct` `\f@tdopunct`

`\glyphpunct` sets punctuation marks in combination with different sorts of letters. The work is done by `\f@tdopunct`.

```
\newcommand*{\glyphpunct}{\par\f@tdopunct{min}\f@tdopunct{pig}\f@tdopunct{hid}
                      \f@tdopunct{HIE}\f@tdopunct{TIP}\f@tdopunct{fluff}
  \$1,234.56 + 7/8 = 9\% @ \#0\par}
\newcommand*{\f@tdopunct}[1]{#1,\ #1:\ #1;\ ‘#1’\
  ?‘#1?\ !‘#1!\ (#1)\ [#1]\ #1*\ #1.\par}
```

The end of the package.

```
⟨/pack⟩
```

Bibliography

[Lik05] Knut Lickert. *Blindtext.sty: Creating text for testing / Texterzeugung zum testen.* October 2005. (Available from CTAN in macros/latex/contrib/blindtext)

[MG04] Frank Mittelbach and Michel Goossens. *The LaTeX Companion.* Second edition. Addison-Wesley Publishing Company, 2004.

[Pak05] Scott Pakin ‘The mftinc package’, January 2005. (Available from CTAN in macros/latex/contrib/mftinc)

Index

Numbers written in italic refer to the page where the corresponding entry is described; numbers underlined refer to the code line of the definition; numbers in roman refer to the code lines where the entry is used.