LaTeX 2ε font selection

Sep 2024

Contents

*Thanks to Arash Esbati for documenting the newer NFSS features of 2020

1 Introduction

This document describes the new font selection features of the LaTeX Document Preparation System. It is intended for package writers who want to write font-loading packages similar to `times` or `latexsym`.

This document is only a brief introduction to the new facilities and is intended for package writers who are familiar with TeX fonts and LaTeX packages. It is *neither* a user-guide *nor* a reference manual for fonts in LaTeX 2_ε.

1.1 LaTeX 2_ε fonts

The most important difference between LaTeX 2.09 and LaTeX 2_ε is the way that fonts are selected. In LaTeX 2.09, the Computer Modern fonts were built into the LaTeX format, and so customizing LaTeX to use other fonts was a major effort.

In LaTeX 2_ε, very few fonts are built into the format, and there are commands to load new text and math fonts. Packages such as `times` or `latexsym` allow authors to access these fonts. This document describes how to write similar font-loading packages.

The LaTeX 2_ε font selection system was first released as the 'New Font Selection Scheme' (NFSS) in 1989, and then in release 2 in 1993. LaTeX 2_ε includes NFSS release 2 as standard.

1.2 Overview

This document contains an overview of the new font commands of LaTeX.

Section 2 describes the commands for selecting fonts in classes and packages. It lists the five LaTeX font attributes, and lists the commands for selecting fonts. It also describes how to customize the author commands such as `\textrm` and `\textit` to suit your document design.

Section 3 explains the commands for controlling LaTeX math fonts. It describes how to specify new math fonts and new math symbols.

Section 4 explains how to install new fonts into LaTeX. It shows how LaTeX font attributes are turned into TeX font names, and how to specify your own fonts using font definition files.

Section 5 discusses text font encodings. It describes how to declare a new encoding and how to define commands, such as `\AE` or `\"`, which have different definitions in different encodings, depending on whether ligatures, etc. are available in the encoding.

Section 6 covers font miscellanea. It describes how LaTeX performs font substitution, how to customize fonts that are preloaded in the LaTeX format, and the naming conventions used in LaTeX font selection.

1.3 Further information

For a general introduction to LaTeX, including the new features of LaTeX 2_ε, you should read *LaTeX: A Document Preparation System*, Leslie Lamport, Addison Wesley, 2nd ed, 1994.

A more detailed description of the LaTeX font selection scheme is to be found in *The LaTeX Companion*, 2nd ed, by Mittelbach and Goossens, Addison Wesley, 2004.

The LaTeX font selection scheme is based on TeX, which is described by its developer in *The TeXbook*, Donald E. Knuth, Addison Wesley, 1986, revised in 1991 to include the features of TeX 3.

Sebastian Rahtz's `psnfss` software contains the software for using a large number of Type 1 fonts (including the Adobe Laser Writer 35 and the Monotype CD-ROM fonts) in LaTeX. It should be available from the same source as your copy of LaTeX.

The `psnfss` software uses fonts generated by Alan Jeffrey's `fontinst` software. This can convert fonts from Adobe Font Metric format into a format readable by LaTeX, including the generation of the font definition files described in Section 4. The `fontinst` software should be available from the same source as your copy of LaTeX.

Whenever practical, LaTeX uses the font naming scheme called 'fontname'; this was described in *Filenames for fonts*,[1] *TUGboat* 11(4), 1990.

[1] An up-to-date electronic version of this document can be found on any CTAN server, in the directory `info/fontname`.

The class-writer's guide *LaTeX 2ε for Class and Package Writers* describes the new LaTeX features for writers of document classes and packages and is kept in `clsguide.tex`. Configuring LaTeX is covered by the guide *Configuration options for LaTeX 2ε* in `cfgguide.tex` whilst the philosophy behind our policy on modifying LaTeX is described in *Modifying LaTeX* in `modguide.tex`.

The documented source code (from the files used to produce the kernel format via `latex.ltx`) is now available as *The LaTeX 2ε Sources*. This very large document also includes an index of LaTeX commands. It can be typeset from the LaTeX file `source2e.tex` in the `base` directory; this uses the class file `ltxdoc.cls`.

For more information about TeX and LaTeX, please contact your local TeX Users Group, or the international TeX Users Group. Addresses and other details can be found at:

`https://www.tug.org/lugs.html`

2 Text fonts

This section describes the commands available to class and package writers for specifying and selecting fonts.

2.1 Text font attributes

Every text font in LaTeX has five *attributes*:

encoding This specifies the order that characters appear in the font. The two most common text encodings used in LaTeX are Knuth's 'TeX text' encoding, and the 'TeX text extended' encoding developed by the TeX Users Group members during a TeX Conference at Cork in 1990 (hence its informal name 'Cork encoding').

family The name for a collection of fonts, usually grouped under a common name by the font foundry. For example, 'Adobe Times', 'ITC Garamond', and Knuth's 'Computer Modern Roman' are all font families.

series How heavy and/or expanded a font is. For example, 'medium weight', 'narrow' and 'bold extended' are all series.

shape The form of the letters within a font family. For example, 'italic', 'oblique' and 'upright' (sometimes called 'roman') are all font shapes.

size The design size of the font, for example '10pt'. If no dimension is specified, 'pt' is assumed.

The possible values for these attributes are given short acronyms by LaTeX. The most common values for the font encoding are:

OT1	TEX text
T1	TEX extended text
OML	TEX math italic
OMS	TEX math symbols
OMX	TEX math large symbols
U	Unknown
L⟨*xx*⟩	A local encoding

The 'local' encodings are intended for font encodings which are only locally available, for example a font containing an organization's logo in various sizes.

There are far too many font families to list them all, but some common ones are:

cmr	Computer Modern Roman
cmss	Computer Modern Sans
cmtt	Computer Modern Typewriter
cmm	Computer Modern Math Italic
cmsy	Computer Modern Math Symbols
cmex	Computer Modern Math Extensions
ptm	Adobe Times
phv	Adobe Helvetica
pcr	Adobe Courier

New description 2019/07/10

The font series is denoting a combination of the weight (boldness) and the width (amount of expansion). The standard supported for weights and widths are:

ul	Ultra Light	uc	Ultra Condensed	50%
el	Extra Light	ec	Extra Condensed	62.5%
l	Light	c	Condensed	75%
sl	Semi Light	sc	Semi Condensed	87.5%
m	Medium (normal)	m	Medium	100%
sb	Semi Bold	sx	Semi Expanded	112.5%
b	Bold	x	Expanded	125%
eb	Extra Bold	ex	Extra Expanded	150%
ub	Ultra Bold	ux	Ultra Expanded	200%

These are concatenated to a single series value except that m is dropped unless both weight and width are medium in which case a single m is used.

Examples for series values are then:

m	Medium weight and width
b	Bold weight, medium width
bx	Bold extended
sb	Semi-bold, medium width
sbx	Semi-bold extended
c	Medium weight, condensed width

Note, that there are a large variety of names floating around like “regular”, “black”, “demi-bold”, “thin”, “heavy” and many more. If at all possible they should be matched into the standard naming scheme to allow for sensible default substitutions if necessary, e.g., “demi-bold” is normally just another name for “semi-bold”, so should get `sb` assigned, etc.

New description 2019/07/10

The most common values for the font shape are:

New description 2020/02/02

`n`	Normal (that is ‘upright’ or ‘roman’)
`it`	Italic
`sl`	Slanted (or ‘oblique’)
`sc`	Caps and small caps
`scit`	Caps and small caps italic
`scsl`	Caps and small caps slanted
`sw`	Swash

A less common value for font shape is:

`ssc`	Spaced caps and small caps

and there is also `ui` for upright italic, i.e., an italic shape but artificially turned upright. This is sometimes useful and available in some fonts.

The font size is specified as a dimension, for example `10pt` or `1.5in` or `3mm`; if no unit is specified, `pt` is assumed. These five parameters specify every LaTeX font, for example:

LaTeX specification	*Font*	*TeX font name*
`OT1 cmr m n 10`	Computer Modern Roman 10 point	`cmr10`
`OT1 cmss m sl 1pc`	Computer Modern Sans Oblique 1 pica	`cmssi12`
`OML cmm m it 10pt`	Computer Modern Math Italic 10 point	`cmmi10`
`T1 ptm b it 1in`	Adobe Times Bold Italic 1 inch	`ptmb8t at 1in`

These five parameters are displayed whenever LaTeX gives an overfull box warning, for example:

```
Overfull \hbox (3.80855pt too wide) in paragraph at lines 314--318
[]\OT1/cmr/m/n/10 Normally [] and [] will be iden-ti-cal,
```

The author commands for fonts set the five attributes as shown in table 1 on the following page. The values used by these commands are determined by the document class, using the parameters defined in Section 2.4.

Note that there are no author commands for selecting new encodings. These should be provided by packages, such as the `fontenc` package.

This section does not explain how LaTeX font specifications are turned into TeX font names. This is described in Section 4.

Author command	*Attribute*	*Value in* `article` *class*
`\textnormal{..}` or `\normalfont`	family	`cmr`
	series	`m`
	shape	`n`
`\textrm{..}` or `\rmfamily`	family	`cmr`
`\textsf{..}` or `\sffamily`	family	`cmss`
`\texttt{..}` or `\ttfamily`	family	`cmtt`
`\textmd{..}` or `\mdseries`	series	`m`
`\textbf{..}` or `\bfseries`	series	`bx`
`\textit{..}` or `\itshape`	shape	`it`
`\textsl{..}` or `\slshape`	shape	`sl`
`\textsc{..}` or `\scshape`	shape	`sc`
`\textssc{..}` or `\sscshape`	shape	`ssc`
`\textsw{..}` or `\swshape`	shape	`sw`
`\textulc{..}` or `\ulcshape`	shape	`ulc` (virtual) → `n`, `it`, `sl` or `ssc`
`\textup{..}` or `\upshape`	shape	`up` (virtual) → `n` or `sc`
`\tiny`	size	5pt
`\scriptsize`	size	7pt
`\footnotesize`	size	8pt
`\small`	size	9pt
`\normalsize`	size	10pt
`\large`	size	12pt
`\Large`	size	14.4pt
`\LARGE`	size	17.28pt
`\huge`	size	20.74pt
`\Huge`	size	24.88pt

Table 1: Author font commands and their effects (article class)

2.2 Selection commands

The low-level commands used to select a text font are as follows.

`\fontencoding {`⟨*encoding*⟩`}`
`\fontfamily {`⟨*family*⟩`}` `\fontseries {`⟨*series*⟩`}` `\fontshape {`⟨*shape*⟩`}`
`\fontsize {`⟨*size*⟩`} {`⟨*baselineskip*⟩`}` `\linespread {`⟨*factor*⟩`}`

New description 1998/12/01

Each of the commands starting with `\font...` sets one of the font attributes; `\fontsize` also sets `\baselineskip`. The `\linespread` command prepares to multiply the current (or newly defined) `\baselineskip` with ⟨*factor*⟩ (e.g., spreads the lines apart for values greater one).

The actual font in use is not altered by these commands, but the current attributes are used to determine which font and baseline skip to use after the next `\selectfont` command.

`\selectfont`

Selects a text font, based on the current values of the font attributes.

Warning: There *must* be a `\selectfont` command immediately after any settings of the font parameters by (some of) the six commands above, before any following text. For example, it is legal to say:

```
\fontfamily{ptm}\fontseries{b}\selectfont Some text.
```

but it is *not* legal to say:

```
\fontfamily{ptm} Some \fontseries{b}\selectfont text.
```

You may get unexpected results if you put text between a `\font`⟨*parameter*⟩ command (or `\linespread`) and a `\selectfont`.

`\usefont {`⟨*encoding*⟩`} {`⟨*family*⟩`} {`⟨*series*⟩`} {`⟨*shape*⟩`}`

A short hand for the equivalent `\font...` commands followed by a call to `\selectfont`.

2.3 Internals

The current values of the font attributes are held in internal macros.

`\f@encoding` `\f@family` `\f@series` `\f@shape` `\f@size` `\f@baselineskip`
`\tf@size` `\sf@size` `\ssf@size`

These hold the current values of the encoding, the family, the series, the shape, the size, the baseline skip, the main math size, the 'script' math size and the 'scriptscript' math size. The last three are accessible only within a formula; outside of math they may contain arbitrary values.

For example, to set the size to 12 without changing the baseline skip:

```
\fontsize{12}{\f@baselineskip}
```

However, you should *never* alter the values of the internal commands directly; they must only be modified using the low-level commands like `\fontfamily`, `\fontseries`, etc. If you disobey this warning you might produce code that loops.

2.4 Parameters for author commands

The parameter values set by author commands such as `\textrm` and `\rmfamily`, etc. are not hard-wired into LaTeX; instead these commands use the values of a number of parameters set by the document class and packages. For example, `\rmdefault` is the name of the default family selected by `\textrm` and `\rmfamily`. Thus to set a document in Adobe Times, Helvetica and Courier, the document designer specifies:

```
\renewcommand{\rmdefault}{ptm}
\renewcommand{\sfdefault}{phv}
\renewcommand{\ttdefault}{pcr}
```

`\encodingdefault` `\familydefault` `\seriesdefault` `\shapedefault`

The encoding, family, series and shape of the main body font. By default these are `OT1`, `\rmdefault`, `m` and `n`. Note that since the default family is `\rmdefault`, this means that changing `\rmdefault` will change the main body font of the document.

`\rmdefault` `\sfdefault` `\ttdefault`

The families selected by `\textrm`, `\rmfamily`, `\textsf`, `\sffamily`, `\texttt` and `\ttfamily`. By default these are `cmr`, `cmss` and `cmtt`.

`\bfdefault` `\mddefault`

The series selected by `\textbf`, `\bfseries`, `\textmd` and `\mdseries`. By default these are `bx` and `m`. These values are suitable for the default families used. If other fonts are used as standard document fonts (for example, certain PostScript fonts) it might be necessary to adjust the value of `\bfdefault` to `b` since only a few such families have a 'bold extended' series. An alternative (taken for the fonts provided by `psnfss`) is to define silent substitutions from `bx` series to `b` series with special `\DeclareFontShape` declarations and the `ssub` size function, see Section 4.4.

`\itdefault` `\sldefault` `\scdefault` `\sscdefault` `\swdefault` `\ulcdefault` `\updefault`

New feature 2020/02/02

The shapes selected by `\textit`, `\itshape`, `\textsl`, `\slshape`, `\textsc`, `\scshape`, `\textssc`, `\sscshape`, `\textsw`, `\swshape`, `\textulc`, `\ulcshape`, `\textup` and `\upshape`. By default these are `it`, `sl`, `sc`, `ssc`, `sw`, `ulc` and `up`. Note that `ulc` and `up` are special here because they are virtual shapes; they don't exist as real shape values. Instead they alter the existing shape value based on rules, i.e., the result depends on context. The respective macros `\textulc` or `\ulcshape` change small capitals back to upper/lower case but will not change the font with respect to italics, slanted or swash. `\upshape` or `\textup` in contrast will switch italics or slanted back to upright but not alter the state of upper/lower case, e.g., keep small capitals if present. Finally, the command `\normalshape` is provided to reset the shape back to normal which is a shorthand for `\upshape\ulcshape`.

Note that there are no parameters for the size commands. These should be defined directly in class files, for example:

```
\renewcommand{\normalsize}{\fontsize{10}{12}\selectfont}
```

More elaborate examples (setting additional parameters when the text size is changed) can be found in `classes.dtx` the source documentation for the classes `article`, `report`, and `book`.

2.5 Special font declaration commands

`\DeclareFixedFont {`⟨*cmd*⟩`} {`⟨*encoding*⟩`} {`⟨*family*⟩`} {`⟨*series*⟩`} {`⟨*shape*⟩`} {`⟨*size*⟩`}`

Declares command ⟨*cmd*⟩ to be a font switch which selects the font that is specified by the attributes ⟨*encoding*⟩, ⟨*family*⟩, ⟨*series*⟩, ⟨*shape*⟩, and ⟨*size*⟩.

The font is selected without any adjustments to baselineskip and other surrounding conditions.

This example makes `{\picturechar .}` select a small dot very quickly:

```
\DeclareFixedFont{\picturechar}{OT1}{cmr}{m}{n}{5}
```

`\DeclareTextFontCommand {`⟨*cmd*⟩`} {`⟨*font-switches*⟩`}`

Declares command ⟨*cmd*⟩ to be a font command with one argument. The current font attributes are locally modified by ⟨*font-switches*⟩ and then the argument of ⟨*cmd*⟩ is typeset in the resulting new font.

Commands defined by `\DeclareTextFontCommand` automatically take care of any necessary italic correction (on either side).

The following example shows how `\textrm` is defined by the kernel.

```
\DeclareTextFontCommand{\textrm}{\rmfamily}
```

To define a command that always typeset its argument in the italic shape of the main document font you could declare:

```
\DeclareTextFontCommand{\normalit}{\normalfont\itshape}
```

This declaration can be used to change the meaning of a command; if ⟨*cmd*⟩ is already defined, a log that it has been redefined is put in the transcript file.

`\DeclareOldFontCommand {`⟨*cmd*⟩`} {`⟨*text-switch*⟩`} {`⟨*math-switch*⟩`}`

Declares command ⟨*cmd*⟩ to be a font switch (i.e. used with the syntax `{`⟨*cmd*⟩`...}`) having the definition ⟨*text-switch*⟩ when used in text and the definition ⟨*math-switch*⟩ when used in a formula. Math alphabet commands, like `\mathit`, when used within ⟨*math-switch*⟩ should not have an argument. Their use in this argument causes their semantics to change so that they here act as a font switch, as required by the usage of the ⟨*cmd*⟩.

This declaration is useful for setting up commands like `\rm` to behave as they did in LaTeX 2.09. We strongly urge you *not* to misuse this declaration to invent new font commands.

The following example defines `\it` to produce the italic shape of the main document font if used in text and to switch to the font that would normally be produced by the math alphabet `\mathit` if used in a formula.

```
\DeclareOldFontCommand{\it}{\normalfont\itshape}{\mathit}
```

This declaration can be used to change the meaning of a command; if ⟨*cmd*⟩ is already defined, a log that it has been redefined is put in the transcript file.

3 Math fonts

This section describes the commands available to class and package writers for specifying math fonts and math commands.

3.1 Math font attributes

The selection of fonts within math mode is quite different to that of text fonts.

Some math fonts are selected explicitly by one-argument commands such as `\mathsf{max}` or `\mathbf{vec}`; such fonts are called *math alphabets*. These math alphabet commands affect only the font used for letters and symbols of type `\mathalpha` (see Section 3.6); other symbols within the argument will be left unchanged. The predefined math alphabets are:

Alphabet	*Description*	*Example*
`\mathnormal`	default	$abcXYZ$
`\mathrm`	roman	abcXYZ
`\mathbf`	bold roman	$\mathbf{abcXYZ}$
`\mathsf`	sans serif	abcXYZ
`\mathit`	text italic	abcXYZ
`\mathtt`	typewriter	$\mathtt{abcXYZ}$
`\mathcal`	calligraphic	$\mathcal{XYZ}$

Other math fonts are selected implicitly by TEX for symbols, with commands such as `\oplus` (producing $\oplus$) or with straight characters like `>` or `+`. Fonts containing such math symbols are called *math symbol fonts*. The predefined math symbol fonts are:

Symbol font	*Description*	*Example*
`operators`	symbols from `\mathrm`	$[+]$
`letters`	symbols from `\mathnormal`	$<<\star>>$
`symbols`	most LATEX symbols	$\leq * \geq$
`largesymbols`	large symbols	$\sum\prod\int$

Some math fonts are both *math alphabets* and *math symbol fonts*, for example `\mathrm` and `operators` are the same font, and `\mathnormal` and `letters` are the same font.

Math fonts in LaTeX have the same five attributes as text fonts: encoding, family, series, shape and size. However, there are no commands that allow the attributes to be individually changed. Instead, the conversion from math fonts to these five attributes is controlled by the *math version*. For example, the `normal` math version maps:

Math font		*External font*
Alphabets	*Symbol fonts*	*Attributes*
`\mathnormal`	`letters`	`OML cmm m it`
`\mathrm`	`operators`	`OT1 cmr m n`
`\mathcal`	`symbols`	`OMS cmsy m n`
	`largesymbols`	`OMX cmex m n`
`\mathbf`		`OT1 cmr bx n`
`\mathsf`		`OT1 cmss m n`
`\mathit`		`OT1 cmr m it`
`\mathtt`		`OT1 cmtt m n`

The `bold` math version is similar except that it contains bold fonts. The command `\boldmath` selects the `bold` math version.

Math versions can only be changed outside of math mode.

The two predefined math versions are:

`normal`	the default math version
`bold`	the bold math version

Packages may define new math alphabets, math symbol fonts, and math versions. This section describes the commands for writing such packages.

3.2 Selection commands

There are no commands for selecting symbol fonts. Instead, these are selected indirectly through symbol commands like `\oplus`. Section 3.6 explains how to define symbol commands.

`\mathnormal{`⟨*math*⟩`}` `\mathcal{`⟨*math*⟩`}` `\mathbf{`⟨*math*⟩`}` `\mathit{`⟨*math*⟩`}`
`\mathrm{`⟨*math*⟩`}` `\mathsf{`⟨*math*⟩`}` `\mathtt{`⟨*math*⟩`}`

Each math alphabet is a command which can only be used inside math mode. For example, `$x + \mathsf{y} + \mathcal{Z}$` produces $x + \mathsf{y} + \mathcal{Z}$.

`\mathversion{`⟨*version*⟩`}`

This command selects a math version; it can only be used outside math mode. For example, `\boldmath` is defined to be `\mathversion{bold}`.

3.3 Declaring math versions

`\DeclareMathVersion {`⟨*version*⟩`}`

Defines ⟨*version*⟩ to be a math version.

The newly declared version is initialized with the defaults for all symbol fonts and math alphabets declared so far (see the commands `\DeclareSymbolFont` and `\DeclareMathAlphabet`).

If used on an already existing version, an information message is written to the transcript file and all previous `\SetSymbolFont` or `\SetMathAlphabet` declarations for this version are overwritten by the math alphabet and symbol font defaults, i.e. one ends up with a virgin math version.

Example:

```
\DeclareMathVersion{normal}
```

3.4 Declaring math alphabets

`\DeclareMathAlphabet {`⟨*math-alph*⟩`} {`⟨*encoding*⟩`} {`⟨*family*⟩`} {`⟨*series*⟩`} {`⟨*shape*⟩`}`

New description 1997/12/01

If this is the first declaration for ⟨*math-alph*⟩ then a new math alphabet with this as its command name is created.

The arguments ⟨*encoding*⟩ ⟨*family*⟩ ⟨*series*⟩ ⟨*shape*⟩ are used to set, or reset, the default values for this math alphabet in all math versions; if required, these must be further reset later for a particular math version by a `\SetMathAlphabet` command.

If ⟨*shape*⟩ is empty then this ⟨*math-alph*⟩ is declared to be invalid in all versions, unless it is set by a later `\SetMathAlphabet` command for a particular math version.

Checks that the command ⟨*math-alph*⟩ is either already a math alphabet command or is undefined; and that ⟨*encoding*⟩ is a known encoding scheme, i.e., has been previously declared.

In these examples, `\foo` is defined for all math versions but `\baz`, by default, is defined nowhere.

```
\DeclareMathAlphabet{\foo}{OT1}{cmtt}{m}{n}
\DeclareMathAlphabet{\baz}{OT1}{}{}{}
```

`\SetMathAlphabet {`⟨*math-alph*⟩`} {`⟨*version*⟩`}`
`{`⟨*encoding*⟩`} {`⟨*family*⟩`} {`⟨*series*⟩`} {`⟨*shape*⟩`}`

Changes, or sets, the font for the math alphabet ⟨*math-alph*⟩ in math version ⟨*version*⟩ to ⟨*encoding*⟩⟨*family*⟩⟨*series*⟩⟨*shape*⟩.

Checks that ⟨*math-alph*⟩ has been declared as a math alphabet, ⟨*version*⟩ is a known math version and ⟨*encoding*⟩ is a known encoding scheme.

This example defines `\baz` for the 'normal' math version only:

```
\SetMathAlphabet{\baz}{normal}{OT1}{cmss}{m}{n}
```

Note that this declaration is not used for all math alphabets: Section 3.5 describes `\DeclareSymbolFontAlphabet`, which is used to set up math alphabets contained in fonts which have been declared as symbol fonts.

3.5 Declaring symbol fonts

`\DeclareSymbolFont {`⟨*sym-font*⟩`} {`⟨*encoding*⟩`} {`⟨*family*⟩`} {`⟨*series*⟩`} {`⟨*shape*⟩`}`

If this is the first declaration for ⟨*sym-font*⟩ then a new symbol font with this name is created (i.e. this identifier is assigned to a new TEX math group).

New description 1997/12/01

The arguments ⟨*encoding*⟩ ⟨*family*⟩ ⟨*series*⟩ ⟨*shape*⟩ are used to set, or reset. the default values for this symbol font in *all* math versions; if required, these must be further reset later for a particular math version by a `\SetSymbolFont` command.

Checks that ⟨*encoding*⟩ is a declared encoding scheme.

For example, the following sets up the first four standard math symbol fonts:

```
\DeclareSymbolFont{operators}{OT1}{cmr}{m}{n}
\DeclareSymbolFont{letters}{OML}{cmm}{m}{it}
\DeclareSymbolFont{symbols}{OMS}{cmsy}{m}{n}
\DeclareSymbolFont{largesymbols}{OMX}{cmex}{m}{n}
```

`\SetSymbolFont {`⟨*sym-font*⟩`} {`⟨*version*⟩`}`
`{`⟨*encoding*⟩`} {`⟨*family*⟩`} {`⟨*series*⟩`} {`⟨*shape*⟩`}`

Changes the symbol font ⟨*sym-font*⟩ for math version ⟨*version*⟩ to ⟨*encoding*⟩ ⟨*family*⟩ ⟨*series*⟩ ⟨*shape*⟩.

Checks that ⟨*sym-font*⟩ has been declared as a symbol font, ⟨*version*⟩ is a known math version and ⟨*encoding*⟩ is a declared encoding scheme.

For example, the following come from the set up of the 'bold' math version:

```
\SetSymbolFont{operators}{bold}{OT1}{cmr}{bx}{n}
\SetSymbolFont{letters}{bold}{OML}{cmm}{b}{it}
```

`\DeclareSymbolFontAlphabet {`⟨*math-alph*⟩`} {`⟨*sym-font*⟩`}`

Allows the previously declared symbol font ⟨*sym-font*⟩ to be the math alphabet with command ⟨*math-alph*⟩ in *all* math versions.

New description 1997/12/01

Checks that the command ⟨*math-alph*⟩ is either already a math alphabet command or is undefined; and that ⟨*sym-font*⟩ is a symbol font.

Example:

```
\DeclareSymbolFontAlphabet{\mathrm}{operators}
\DeclareSymbolFontAlphabet{\mathcal}{symbols}
```

This declaration should be used in preference to `\DeclareMathAlphabet` and `\SetMathAlphabet` when a math alphabet is the same as a symbol font; this is because it makes better use of the limited number (only 16) of TEX's math groups.

Note that, whereas a TEX math group is allocated to each symbol font when it is first declared, a math alphabet uses a TEX math group only when its command is used within a math formula.

New description 1997/12/01

3.6 Declaring math symbols

`\DeclareMathSymbol {`⟨*symbol*⟩`} {`⟨*type*⟩`} {`⟨*sym-font*⟩`} {`⟨*slot*⟩`}`

The ⟨*symbol*⟩ can be either a single character such as '`>`', or a macro name, such as `\sum`.

Defines the ⟨*symbol*⟩ to be a math symbol of type ⟨*type*⟩ in slot ⟨*slot*⟩ of symbol font ⟨*sym-font*⟩. The ⟨*type*⟩ can be given as a number or as a command:

Type	*Meaning*	*Example*
0 or `\mathord`	Ordinary	α
1 or `\mathop`	Large operator	$\sum$
2 or `\mathbin`	Binary operation	$\times$
3 or `\mathrel`	Relation	$\leq$
4 or `\mathopen`	Opening	$\langle$
5 or `\mathclose`	Closing	$\rangle$
6 or `\mathpunct`	Punctuation	;
7 or `\mathalpha`	Alphabet character	A

Only symbols of type `\mathalpha` will be affected by math alphabet commands: within the argument of a math alphabet command they will produce the character in slot ⟨*slot*⟩ of that math alphabet's font. Symbols of other types will always produce the same symbol (within one math version).

`\DeclareMathSymbol` allows a macro ⟨*symbol*⟩ to be redefined only if it was previously defined to be a math symbol. It also checks that the ⟨*sym-font*⟩ is a declared symbol font.

Example:

```
\DeclareMathSymbol{\alpha}{0}{letters}{"0B}
\DeclareMathSymbol{\lessdot}{\mathbin}{AMSb}{"0C}
\DeclareMathSymbol{\alphld}{\mathalpha}{AMSb}{"0C}
```

`\DeclareMathDelimiter {`⟨*cmd*⟩`} {`⟨*type*⟩`} {`⟨*sym-font-1*⟩`} {`⟨*slot-1*⟩`} {`⟨*sym-font-2*⟩`} {`⟨*slot-2*⟩`}`

Defines ⟨*cmd*⟩ to be a math delimiter where the small variant is in slot ⟨*slot-1*⟩ of symbol font ⟨*sym-font-1*⟩ and the large variant is in slot ⟨*slot-2*⟩ of symbol font ⟨*sym-font-2*⟩. Both symbol fonts must have been declared previously.

Checks that ⟨*sym-font-i*⟩ are both declared symbol fonts.

If TEX is not looking for a delimiter, ⟨*cmd*⟩ is treated just as if it had been defined with `\DeclareMathSymbol` using ⟨*type*⟩, ⟨*sym-font-1*⟩ and ⟨*slot-1*⟩. In other words, if a command is defined as a delimiter then this automatically defines it as a math symbol.

In case ⟨*cmd*⟩ is a single character such as '`[`', the same syntax is used. Previously the `{`⟨*type*⟩`}` argument was not present (and thus the corresponding math symbol declaration had to be provided separately).

New description 1998/06/01

Example:

```
\DeclareMathDelimiter{\langle}{\mathopen}{symbols}{"68}
                                         {largesymbols}{"0A}
\DeclareMathDelimiter{(}      {\mathopen}{operators}{"28}
                                         {largesymbols}{"00}
```

`\DeclareMathAccent {`⟨*cmd*⟩`} {`⟨*type*⟩`} {`⟨*sym-font*⟩`} {`⟨*slot*⟩`}`

Defines ⟨*cmd*⟩ to act as a math accent.

The accent character comes from slot ⟨*slot*⟩ in ⟨*sym-font*⟩. The ⟨*type*⟩ can be either `\mathord` or `\mathalpha`; in the latter case the accent character changes font when used in a math alphabet.

Example:

```
\DeclareMathAccent{\acute}{\mathalpha}{operators}{"13}
\DeclareMathAccent{\vec}{\mathord}{letters}{"7E}
```

`\DeclareMathRadical {`⟨*cmd*⟩`} {`⟨*sym-font-1*⟩`} {`⟨*slot-1*⟩`} {`⟨*sym-font-2*⟩`} {`⟨*slot-2*⟩`}`

Defines ⟨*cmd*⟩ to be a radical where the small variant is in slot ⟨*slot-1*⟩ of symbol font ⟨*sym-font-1*⟩ and the large variant is in slot ⟨*slot-2*⟩ of symbol font ⟨*sym-font-2*⟩. Both symbol fonts must have been declared previously.

Example (probably the only use for it!):

```
\DeclareMathRadical{\sqrt}{symbols}{"70}{largesymbols}{"70}
```

3.7 Declaring math sizes

`\DeclareMathSizes {`⟨*t-size*⟩`} {`⟨*mt-size*⟩`} {`⟨*s-size*⟩`} {`⟨*ss-size*⟩`}`

Declares that ⟨*mt-size*⟩ is the (main) math text size, ⟨*s-size*⟩ is the 'script' size and ⟨*ss-size*⟩ the 'scriptscript' size to be used in math, when ⟨*t-size*⟩ is

the current text size. For text sizes for which no such declaration is given the 'script' and 'scriptscript' size will be calculated and then fonts are loaded for the calculated sizes or the best approximation (this may result in a warning message).

Normally, ⟨*t-size*⟩ and ⟨*mt-size*⟩ will be identical; however, if, for example, PostScript text fonts are mixed with bit-map math fonts then you may not have available a ⟨*mt-size*⟩ for every ⟨*t-size*⟩.

Example:

```
\DeclareMathSizes{13.82}{14.4}{10}{7}
```

4 Font installation

This section explains how LaTeX's font attributes are turned into TeX font specifications.

4.1 Font definition files

New description 1997/12/01

The description of how LaTeX font attributes are turned into TeX fonts is usually kept in a *font definition file* (`.fd`). The file for family ⟨*family*⟩ in encoding ⟨*ENC*⟩ must be called ⟨*enc*⟩⟨*family*⟩`.fd`: for example, `ot1cmr.fd` for Computer Modern Roman with encoding `OT1` or `t1ptm.fd` for Adobe Times with encoding `T1`. Note that encoding names are converted to lowercase when used as part of file names.

Whenever LaTeX encounters an encoding/family combination that it does not know (e.g. if the document designer says `\fontfamily{ptm}\selectfont`) then LaTeX attempts to load the appropriate `.fd` file. "Not known" means: there was no `\DeclareFontFamily` declaration issued for this encoding/family combination. If the `.fd` file could not be found, a warning is issued and font substitutions are made.

The declarations in the font definition file are responsible for telling LaTeX how to load fonts for that encoding/family combination.

4.2 Font definition file commands

Note: A font definition file should contain only commands from this subsection.

Note that these commands can also be used outside a font definition file: they can be put in package or class files, or even in the preamble of a document.

`\ProvidesFile{`⟨*file-name*⟩`}[`⟨*release-info*⟩`]`

The file should announce itself with a `\ProvidesFile` command, as described in *LaTeX 2ε for Class and Package Writers*.

For example:

```
\ProvidesFile{t1ptm.fd}[1994/06/01 Adobe Times font definitions]
```

Spaces within the arguments specific to font definition files are ignored to avoid surplus spaces in the document. If a real space is necessary use `\space`. However, note that this is only true if the declaration is made at top level! If used within the definition of another command, within `\AtBeginDocument`, option code or in similar places, then spaces within the argument will remain and may result in incorrect table entries.

New description 2004/02/10

`\DeclareFontFamily {`⟨*encoding*⟩`} {`⟨*family*⟩`} {`⟨*loading-settings*⟩`}`

Declares a font family ⟨*family*⟩ to be available in encoding scheme ⟨*encoding*⟩.

The ⟨*loading-settings*⟩ are executed immediately after loading any font with this encoding and family.

Checks that ⟨*encoding*⟩ was previously declared.

This example refers to the Computer Modern Typewriter font family in the Cork encoding:

```
\DeclareFontFamily{T1}{cmtt}{\hyphenchar\font=-1}
```

Each `.fd` file should contain exactly one `\DeclareFontFamily` command, and it should be for the appropriate encoding/family combination.

`\DeclareFontShape {`⟨*encoding*⟩`} {`⟨*family*⟩`} {`⟨*series*⟩`} {`⟨*shape*⟩`}`
`{`⟨*loading-info*⟩`} {`⟨*loading-settings*⟩`}`

Declares a font shape combination; here ⟨*loading-info*⟩ contains the information that combines sizes with external fonts. The syntax is complex and is described in Section 4.3 below.

The ⟨*loading-settings*⟩ are executed after loading any font with this font shape. They are executed immediately after the 'loading-settings' which were declared by `\DeclareFontFamily` and so they can be used to overwrite the settings made at the family level.

Checks that the combination ⟨*encoding*⟩⟨*family*⟩ was previously declared via `\DeclareFontFamily`.

Example:

```
\DeclareFontShape{OT1}{cmr}{m}{sl}{%
          <5-8> sub * cmr/m/n
          <8> cmsl8
          <9> cmsl9
          <10> <10.95> cmsl10
          <12> <14.4> <17.28> <20.74> <24.88> cmsl12
          }{}
```

The file can contain any number of `\DeclareFontShape` commands, which should be for the appropriate ⟨*encoding*⟩ and ⟨*family*⟩.

The font family declarations for the `OT1`-encoded fonts now all contain: New feature 1996/06/01

```
\hyphenchar\font=‘\-
```

This enables the use of an alternative `\hyphenchar` in other encodings whilst maintaining the correct value for all fonts.

According to NFSS conventions the series value should be a combination of weight and width abbreviated each with one or two letters as described on page 5. In particular it should not contain an “`m`” unless it consists of just one character. In the past incorrect values such as “`cm`” were simply accepted, but since this now leads to problems with the extended mechanism, the correct syntax is now enforced. New feature 2020/02/02

More exactly, if the series values is a member of a specific set of values (`ulm`, `elm`, `lm`, `slm`, `mm`, `sbm`, `bm`, `ebm`, `ubm`, `muc`, `mec`, `mc`, `msc`, `msx`, `mx`, `mex` or `mux`) it is assumed to be in incorrect NFSS notation and so a warning is given and a surplus “`m`” is dropped. Other values are not touched to allow for the usage of values like “`semibold`” or “`medium`” as used by the `autoinst` program.

4.3 Font file loading information

The information which tells LaTeX exactly which font (`.tfm`) files to load is contained in the ⟨*loading-info*⟩ part of a `\DeclareFontShape` declaration. This part consists of one or more ⟨*fontshape-decl*⟩s, each of which has the following form:

⟨*fontshape-decl*⟩ ::= ⟨*size-infos*⟩ ⟨*font-info*⟩
⟨*size-infos*⟩ ::= ⟨*size-infos*⟩ ⟨*size-info*⟩ | ⟨*size-info*⟩
⟨*size-info*⟩ ::= “`<`” ⟨*number-or-range*⟩ “`>`”
⟨*font-info*⟩ ::= [⟨*size-function*⟩ “`*`”] [“`[`” ⟨*optarg*⟩ “`]`”] ⟨*fontarg*⟩

The ⟨*number-or-range*⟩ denotes the size or size-range for which this entry applies.

If it contains a hyphen it is a range: lower bound on the left (if missing, zero implied), upper bound on the right (if missing, ∞ implied). For ranges, the upper bound is *not* included in the range and the lower bound is.

Examples:

`<10>`	simple size	10pt only
`<-8>`	range	all sizes less than 8pt
`<8-14.4>`	range	all sizes greater than or equal to 8pt but less than 14.4pt
`<14.4->`	range	all sizes greater than or equal 14.4pt

If more than one ⟨*size-info*⟩ entry follows without any intervening ⟨*font-info*⟩, they all share the next ⟨*font-info*⟩.

The ⟨*size-function*⟩, if present, handles the use of ⟨*font-info*⟩. If not present, the 'empty' ⟨*size-function*⟩ is assumed.

All the ⟨*size-info*⟩s are inspected in the order in which they appear in the font shape declaration. If a ⟨*size-info*⟩ matches the requested size, its ⟨*size-function*⟩ is executed. If `\external@font` is non-empty afterwards this process stops, otherwise the next ⟨*size-info*⟩ is inspected. (See also `\DeclareSizeFunction`.)

If this process does not lead to a non-empty `\external@font`, LaTeX tries the nearest simple size. If the entry contains only ranges an error is returned.

4.4 Size functions

LaTeX provides the following size functions, whose 'inputs' are ⟨*fontarg*⟩ and ⟨*optarg*⟩ (when present).

'' (empty) Load the external font ⟨*fontarg*⟩ at the user-requested size. If ⟨*optarg*⟩ is present, it is used as the scale-factor.

s Like the empty function but without terminal warnings, only loggings.

gen Generates the external font from ⟨*fontarg*⟩ followed by the user-requested size, e.g. `<8> <9> <10> gen * cmtt`

sgen Like the 'gen' function but without terminal warnings, only loggings.

genb Generates the external font from ⟨*fontarg*⟩ followed by the user-requested size, using the conventions of the 'ec' fonts. e.g. `<10.98> genb * dctt` produces `dctt1098`. New feature 1995/12/01

sgenb Like the 'genb' function but without terminal warnings, only loggings. New feature 1995/12/01

sub Tries to load a font from a different font shape declaration given by ⟨*fontarg*⟩ in the form ⟨*family*⟩/⟨*series*⟩/⟨*shape*⟩.

ssub Silent variant of 'sub', only loggings.

alias Same as 'ssub' but with a different logging message. Intended for cases where the substitution is only done to change the name, e.g., going from `regular` series to the official name `m`. In that case given a warning that some shape is not found is not correct. New feature 2019/10/15

subf Like the empty function but issues a warning that it has to substitute the external font ⟨*fontarg*⟩ because the desired font shape was not available in the requested size.

ssubf Silent variant of 'subf', only loggings.

fixed Load font ⟨*fontarg*⟩ as is, disregarding the user-requested size. If present, ⟨*optarg*⟩ gives the "at ... pt" size to be used.

sfixed Silent variant of 'fixed', only loggings.

Examples for the use of most of the above size functions can be found in the file `cmfonts.fdd`—the source for the standard `.fd` files describing the Computer Modern fonts by Donald Knuth.

`\DeclareSizeFunction {`⟨*name*⟩`} {`⟨*code*⟩`}`

Declares a size-function ⟨*name*⟩ for use in `\DeclareFontShape` commands. The interface is still under development but there should be no real need to a define new size functions.

The ⟨*code*⟩ is executed when the size or size-range in `\DeclareFontShape` matches the user-requested size.

The arguments of the size-function are automatically parsed and placed into `\mandatory@arg` and `\optional@arg` for use in ⟨*code*⟩. Also available, of course, is `\f@size`, which is the user-requested size.

To signal success ⟨*code*⟩ must define the command `\external@font` to contain the external name and any scaling options (if present) for the font to be loaded.

This example sets up the 'empty' size function (simplified):

```
\DeclareSizeFunction{}
        {\edef\external@font{\mandatory@arg\space at\f@size}
```

5 Encodings

This section explains how to declare and use new font encodings and how to declare commands for use with particular encodings.

5.1 The fontenc package

Users can select new font encodings using the `fontenc` package. The `fontenc` package has options for encodings; the last option becomes the default encoding. For example, to use the `OT2` (Washington University Cyrillic encoding) and `T1` encodings, with `T1` as the default, an author types:

```
\usepackage[OT2,T1]{fontenc}
```

New description 1997/12/01

For each font encoding ⟨*ENC*⟩ given as an option, this package loads the *encoding definition* (⟨*enc*⟩`enc.def`, with an all lower-case name) file; it also sets `\encodingdefault` to be the last encoding in the option list.

The declarations in the encoding definition file ⟨*enc*⟩`enc.def` for encoding ⟨*ENC*⟩ are responsible for declaring this encoding and telling LaTeX how to produce characters in this encoding; this file should contain nothing else (see Section 5.2.

The standard LaTeX format declares the `OT1` and `T1` text encodings by inputting the files `ot1enc.def` and `t1enc.def`; it also sets up various defaults which

require that `OT1`-encoded fonts are available. Other encoding set-ups might be added to the distribution at a later stage.

Thus the example above loads the files `ot2enc.def` and `t1enc.def` and sets `\encodingdefault` to `T1`.

Warning: If you wish to use `T1`-encoded fonts other than the 'cmr' family then you may need to load the package (e.g. `times`) that selects the fonts *before* loading `fontenc` (this prevents the system from attempting to load any `T1`-encoded fonts from the 'cmr' family).

5.2 Encoding definition file commands

Note: An encoding definition file should contain only commands from this subsection.

As an exception it may also contain a `\DeclareFontSubstitution` declaration (described in 5.4) to specify how font substitution for this encoding should be handled. In that case it is important that the values used point to a font that is guaranteed to be available on all LaTeX installations.[2]

New description 2019/07/10

As with the font definition file commands, it is also possible (although normally not necessary) to use these declarations directly within a class or package file.

New description 1997/12/01

Warning: Some aspects of the contents of font definition files are still under development. Therefore, the current versions of the files `ot1enc.def` and `t1enc.def` are temporary versions and should not be used as models for producing further such files. For further information you should read the documentation in `ltoutenc.dtx`.

`\ProvidesFile{`⟨*file-name*⟩`}[`⟨*release-info*⟩`]`

The file should announce itself with a `\ProvidesFile` command, described in *LaTeX 2ε for Class and Package Writers*. For example:

```
\ProvidesFile{ot2enc.def}
              [1994/06/01 Washington University Cyrillic encoding]
```

`\DeclareFontEncoding {`⟨*encoding*⟩`} {`⟨*text-settings*⟩`} {`⟨*math-settings*⟩`}`

Declares a new encoding scheme ⟨*encoding*⟩.

The ⟨*text-settings*⟩ are declarations which are executed every time `\selectfont` changes the encoding to be ⟨*encoding*⟩.

The ⟨*math-settings*⟩ are similar but are for math alphabets. They are executed whenever a math alphabet with this encoding is called.

It also saves the value of ⟨*encoding*⟩ in the macro `\LastDeclaredEncoding`.

New feature 1998/12/01

[2] If the font encoding file is made available as part of a CTAN bundle, that could be a font that is provided together with that bundle, but it should not point to font which requires further installation steps and therefore may or may not be installed.

Example:

```
\DeclareFontEncoding{OT1}{}{}
```

Fonts in encoding TS1 are usually not implementing the full encoding but only a subset. This subset should be declared with a \DeclareEncodingSubset declaration:

New feature 2021/06/01

`\DeclareEncodingSubset {`⟨*encoding*⟩`} {`⟨*font family*⟩`} {`⟨*subset number*⟩`}`

This should even be done if the font is implementing the full TS1 encoding; see page 36 for further details.

Some author commands need to change their definition depending on which encoding is currently in use. For example, in the OT1 encoding, the letter 'Æ' is in slot "1D, whereas in the T1 encoding it is in slot "C6. So the definition of \AE has to change depending on whether the current encoding is OT1 or T1. The following commands allow this to happen.

`\DeclareTextCommand {`⟨*cmd*⟩`} {`⟨*encoding*⟩`} [`⟨*num*⟩`] [`⟨*default*⟩`] {`⟨*definition*⟩`}`

This command is like \newcommand, except that it defines a command which is specific to one encoding. For example, the definition of \k in the T1 encoding is:

```
\DeclareTextCommand{\k}{T1}[1]
   {\hmode@bgroup\ooalign{\null#1\crcr\hidewidth\char12}\egroup}
```

\DeclareTextCommand takes the same optional arguments as \newcommand.

The resulting command is robust, even if the code in ⟨*definition*⟩ is fragile.

It does not produce an error if the command has already been defined but logs the redefinition in the transcript file.

`\ProvideTextCommand {`⟨*cmd*⟩`} {`⟨*encoding*⟩`} [`⟨*num*⟩`] [`⟨*default*⟩`] {`⟨*definition*⟩`}`

New feature 1994/12/01

This command is the same as \DeclareTextCommand, except that if ⟨*cmd*⟩ is already defined in encoding ⟨*encoding*⟩, then the definition is ignored.

`\DeclareTextSymbol {`⟨*cmd*⟩`} {`⟨*encoding*⟩`} {`⟨*slot*⟩`}`

This command defines a text symbol with slot ⟨*slot*⟩ in the encoding. For example, the definition of \ss in the OT1 encoding is:

```
\DeclareTextSymbol{\ss}{OT1}{25}
```

It does not produce an error if the command has already been defined but logs the redefinition in the transcript file.

`\DeclareTextAccent {`⟨*cmd*⟩`} {`⟨*encoding*⟩`} {`⟨*slot*⟩`}`

This command declares a text accent, with the accent taken from slot ⟨*slot*⟩ in the encoding. For example, the definition of `\"` in the `OT1` encoding is:

```
\DeclareTextAccent{\"}{OT1}{127}
```

It does not produce an error if the command has already been defined but logs the redefinition in the transcript file.

`\DeclareTextComposite {`⟨*cmd*⟩`} {`⟨*encoding*⟩`} {`⟨*letter*⟩`} {`⟨*slot*⟩`}`

This command declares that the composite letter formed from applying ⟨*cmd*⟩ to ⟨*letter*⟩ is defined to be simply slot ⟨*slot*⟩ in the encoding. The ⟨*letter*⟩ should be a single letter (such as `a`) or a single command (such as `\i`).

For example, the definition of `\'{a}` in the `T1` encoding could be declared like this:

```
\DeclareTextComposite{\'}{T1}{a}{225}
```

The ⟨*cmd*⟩ will normally have been previously declared for this encoding, either by using `\DeclareTextAccent`, or as a one-argument `\DeclareTextCommand`.

`\DeclareTextCompositeCommand {`⟨*cmd*⟩`} {`⟨*encoding*⟩`} {`⟨*letter*⟩`} {`⟨*definition*⟩`}`

New feature 1994/12/01

This is a more general form of `\DeclareTextComposite`, which allows for an arbitrary ⟨*definition*⟩, not just a ⟨*slot*⟩. The main use for this is to allow accents on `i` to act like accents on `\i`, for example:

```
\DeclareTextCompositeCommand{\'}{OT1}{i}{\'\i}
```

It has the same restrictions as `\DeclareTextComposite`.

`\LastDeclaredEncoding`

New feature 1998/12/01

This holds the name of the last encoding declared via `\DeclareFontEncoding` (this should also be the currently most efficient encoding). It can be used in the ⟨*encoding*⟩ argument of the above declarations in place of explicitly mentioning the encoding, e.g.

```
\DeclareFontEncoding{T1}{}{}
\DeclareTextAccent{\`}{\LastDeclaredEncoding}{0}
\DeclareTextAccent{\'}{\LastDeclaredEncoding}{1}
```

This can be useful in cases where encoding files sharing common code are generated from one source.

5.3 Default definitions

New description 1997/12/01

The declarations used in encoding definition files define encoding-specific commands but they do not allow those commands to be used without explicitly changing the encoding. For some commands, such as symbols, this is not enough. For example, the `OMS` encoding contains the symbol '§', but we need to be able to use the command `\S` whatever the current encoding may be, without explicitly selecting the encoding `OMS`.

New description 1997/12/01

To allow this, LaTeX has commands that declare default definitions for commands; these defaults are used when the command is not defined in the current encoding. For example, the default encoding for `\S` is `OMS`, and so in an encoding (such as `OT1`) which does not contain `\S`, the `OMS` encoding is selected in order to access this glyph. But in an encoding (such as `T1`) which does contain `\S`, the glyph in that encoding is used. The standard LaTeX 2_ε format sets up several such defaults using the following encodings: `OT1`, `OMS` and `OML`.

Warning: These commands should *not* occur in encoding definition files, since those files should declare only commands for use when that encoding has been selected. They should instead be placed in packages; they must, of course, always refer to encodings that are known to be available.

`\DeclareTextCommandDefault {`⟨*cmd*⟩`} {`⟨*definition*⟩`}`

New feature 1994/12/01

This command allows an encoding-specific command to be given a default definition. For example, the default definition for `\copyright` is defined be a circled 'c' with:

```
\DeclareTextCommandDefault{\copyright}{\textcircled{c}}
```

`\DeclareTextAccentDefault {`⟨*cmd*⟩`} {`⟨*encoding*⟩`}`
`\DeclareTextSymbolDefault {`⟨*cmd*⟩`} {`⟨*encoding*⟩`}`

New feature 1994/12/01

These commands allow an encoding-specific command to be given a default encoding. For example, the default encoding for `\"` and `\ae` is set to be `OT1` by:

```
\DeclareTextAccentDefault{\"}{OT1}
\DeclareTextSymbolDefault{\ae}{OT1}
```

Note that `\DeclareTextAccentDefault` can be used on any one-argument encoding-specific command, not just those defined with `\DeclareTextAccent`. Similarly, `\DeclareTextSymbolDefault` can be used on any encoding-specific command with no arguments, not just those defined with `\DeclareTextSymbol`.

For more examples of these definitions, see `ltoutenc.dtx`.

`\ProvideTextCommandDefault {⟨cmd⟩} {⟨definition⟩}`

New feature 1994/12/01

This command is the same as `\DeclareTextCommandDefault`, except that if the command already has a default definition, then the definition is ignored. This is useful to give 'faked' definitions of symbols which may be given 'real' definitions by other packages. For example, a package might give a fake definition of `\textonequarter` by saying:

```
\ProvideTextCommandDefault{\textonequarter}{$\m@th\frac14$}
```

5.4 Encoding defaults

`\DeclareFontEncodingDefaults {⟨text-settings⟩} {⟨math-settings⟩}`

Declares ⟨*text-settings*⟩ and ⟨*math-settings*⟩ for all encoding schemes. These are executed before the encoding scheme dependent ones are executed so that one can use the defaults for the major cases and overwrite them if necessary using `\DeclareFontEncoding`.

If `\relax` is used as an argument, the current setting of this default is left unchanged.

This example is used by amsfonts.sty for accent positioning; it changes only the math settings:

```
\DeclareFontEncodingDefaults{\relax}{\def\accentclass@{7}}
```

`\DeclareFontSubstitution {⟨encoding⟩} {⟨family⟩} {⟨series⟩} {⟨shape⟩}`

Declares the default values for font substitution which will be used when a font with encoding ⟨*encoding*⟩ should be loaded but no font can be found with the current attributes.

These substitutions are local to the encoding scheme because the encoding scheme is never substituted! They are tried in the order ⟨*shape*⟩ then ⟨*series*⟩ and finally ⟨*family*⟩.

New description 2019/07/10

This declaration is normally done in an encoding definition file (see 5.2), but can also be used in a class file or the document preamble to alter the default for a specific encoding.

If no defaults are set up for an encoding, the values given by `\DeclareErrorFont` are used.

The font specification for ⟨*encoding*⟩⟨*family*⟩⟨*series*⟩⟨*shape*⟩ must have been defined by `\DeclareFontShape` before the `\begin{document}` is reached.

Example:

```
\DeclareFontSubstitution{T1}{cmr}{m}{n}
```

5.5 Case changing

`\MakeUppercase {`⟨*text*⟩`}`
`\MakeLowercase {`⟨*text*⟩`}`

TEX provides the two primitives `\uppercase` and `\lowercase` for changing the case of text. Unfortunately, these TEX primitives do not change the case of characters accessed by commands like `\ae` or `\aa`. To overcome this problem, LATEX provides these two commands. New feature 1995/06/01

In the long run, we would like to use all-caps fonts rather than any command like `\MakeUppercase` but this is not possible at the moment because such fonts do not exist.

For further details, see `clsguide.tex`.

In order that upper/lower-casing will work reasonably well, and in order to provide any correct hyphenation, LATEX 2ε *must* use, throughout a document, the same fixed table for changing case. The table used is designed for the font encoding `T1`; this works well with the standard TEX fonts for all Latin alphabets but will cause problems when using other alphabets. As an experiment, it has now been extended for use with some Cyrillic encodings. New description 1999/04/23

6 Miscellanea

This section covers the remaining font commands in LATEX and some other issues.

6.1 Font substitution

`\DeclareErrorFont {`⟨*encoding*⟩`} {`⟨*family*⟩`} {`⟨*series*⟩`} {`⟨*shape*⟩`} {`⟨*size*⟩`}`

Declares ⟨*encoding*⟩⟨*family*⟩⟨*series*⟩⟨*shape*⟩ to be the font shape used in cases where the standard substitution mechanism fails (i.e. would loop). For the standard mechanism see the command `\DeclareFontSubstitution` above.

The font specification for ⟨*encoding*⟩⟨*family*⟩⟨*series*⟩⟨*shape*⟩ must have been defined by `\DeclareFontShape` before the `\begin{document}` is reached.

Example:

```
\DeclareErrorFont{OT1}{cmr}{m}{n}{10}
```

This declaration is a system wide fallback and it should normally not be changed, in particular it does not belong into font encoding definition files but rather into the LATEX format. It is normally set up in `fonttext.cfg`. Adjustments on a per encoding base should be made through `\DeclareFontSubstitution` instead! New description 2019/07/10

`\fontsubfuzz`

This parameter is used to decide whether or not to produce a terminal warning if a font size substitution takes place. If the difference between the requested and the chosen size is less than `\fontsubfuzz` the warning is only written to the transcript file. The default value is `0.4pt`. This can be redefined with `\renewcommand`, for example:

```
\renewcommand{\fontsubfuzz}{0pt}   % always warn
```

6.2 Preloading

`\DeclarePreloadSizes {`⟨*encoding*⟩`} {`⟨*family*⟩`} {`⟨*series*⟩`} {`⟨*shape*⟩`} {`⟨*size-list*⟩`}`

Specifies the fonts that should be preloaded by the format. These commands should be put in a `preload.cfg` file, which is read in when the LaTeX format is being built. Read `preload.dtx` for more information on how to built such a configuration file.

Example:

```
\DeclarePreloadSizes{OT1}{cmr}{m}{sl}{10,10.95,12}
```

New description 2019/07/10

Preloading is really an artifact of the days when loading fonts while processing a document contributed substantially to the processing time. These days it is usually best not to use this mechanism any more.

6.3 Accented characters

New description 1996/06/01

Accented characters in LaTeX can be produced using commands such as `\"a` etc. The precise effect of such commands depends on the font encoding being used. When using a font encoding that contains the accented characters as individual glyphs (such as the `T1` encoding, in the case of `\"a`) words that contain such accented characters can be automatically hyphenated. For font encodings that do not contain the requested individual glyph (such as the `OT1` encoding) such a command invokes typesetting instructions that produce the accented character as a combination of character glyphs and diacritical marks in the font. In most cases this involves a call to the TeX primitive `\accent`. Glyphs constructed as composites in this way inhibit hyphenation of the current word; this is one reason why the `T1` encoding is preferable to the original TeX font encoding `OT1`.

It is important to understand that commands like `\"a` in LaTeX 2_ε represent just a name for a single glyph (in this case 'umlaut a') and contain no information about how to typeset that glyph—thus it does *not* mean 'put two dots on top of the character a'. The decision as to what typesetting routine to use will depend on the encoding of the current font and so this decision is taken at the last minute. Indeed, it is possible that the same input will be typeset in more than one way in the same document; for example, text in section headings may also

appear in table of contents and in running heads; and each of these may use a font with a different encoding.

For this reason the notation `\"a` is *not* equivalent to:

```
\newcommand \chara {a}     \"\chara
```

In the latter case, LaTeX does not expand the macro `\chara` but simply compares the notation (the string `\"\chara`) to its list of known composite notations in the current encoding; when it fails to find `\"\chara` it does the best it can and invokes the typesetting instructions that put the umlaut accent on top of the expansion of `\chara`. Thus, even if the font actually contains 'ä' as an individual glyph, it will not be used.

The low-level accent commands in LaTeX are defined in such a way that it is possible to combine a diacritical mark from one font with a glyph from another font; for example, `\"\textparagraph` will produce ¶̈. The umlaut here is taken from the `OT1` encoded font `cmr10` whilst the paragraph sign is from the `OMS` encoded font `cmsy10`. (This example may be typographically silly but better ones would involve font encodings like `OT2` (Cyrillic) that might not be available at every site.)

There are, however, restrictions on the font-changing commands that will work within the argument to such an accent command. These are TeXnical in the sense that they follow from the way that TeX's `\accent` primitive works, allowing only a special class of commands between the accent and the accented character.

The following are examples of commands that will not work correctly as the accent will appear above a space: the font commands with text arguments (`\textbf{...}` and friends); all the font size declarations (`\fontsize` and `\Large`, etc.); `\usefont` and declarations that depend on it, such as `\normalfont`; box commands (e.g. `\mbox{...}`).

The lower-level font declarations that set the attributes family, series and shape (such as `\fontshape{sl}\selectfont`) will produce correct typesetting, as will the default declarations such as `\bfseries`.

6.4 Naming conventions

- Math alphabet commands all start with `\math...`: examples are `\mathbf`, `\mathcal`, etc.
- The text font changing commands with arguments all start with `\text...`: e.g. `\textbf` and `\textrm`. The exception to this is `\emph`, since it occurs very commonly in author documents and so deserves a shorter name.
- Names for encoding schemes are strings of up to three letters (all upper case) plus digits.

  The LaTeX Project reserves the use of encodings starting with the following letters: `T` (standard 256-long text encodings), `TS` (symbols that are designed to extend a corresponding `T` encoding), `X` (text encodings that

do not conform to the strict requirements for `T` encodings), `M` (standard 256-long math encodings), `S` (other symbol encodings), `A` (other special applications), `OT` (standard 128-long text encodings) and `OM` (standard 128-long math encodings).

Please do not use the above starting letters for non-portable encodings. If new standard encodings emerge then we shall add them in a later release of LaTeX.

Encoding schemes which are local to a site or a system should start with `L`, experimental encodings intended for wide distribution will start with `E`, whilst `U` is for Unknown or Unclassified encodings.

- Font family names should contain only upper and lower case letters and hyphen characters. Where possible, these should conform to the *Filenames for fonts* font naming scheme of the scheme implemented by `autoinst` with suffixes such as `-LF`, `-OsF`, etc. to indicate different figure styles. New description 2019/10/15
- Font series names should contain up to four lower case letters. If at all possible standard names as suggested in Section 2.1 should be used. Font specific names such as `regular` or `black`, etc. should be at least aliased to a corresponding standard name. New description 2019/10/15
- Font shapes should contain up to four letters lower case. Use the names suggested in Section 2.1. New description 2019/10/15
- Names for symbol fonts are built from lower and upper case letters with no restriction.

Whenever possible, you should use the series and shape names suggested in *The LaTeX Companion* since this will make it easier to combine new fonts with existing fonts.

Where possible, text symbols should be named as `\text` followed by the Adobe glyph name: for example `\textonequarter` or `\textsterling`. Similarly, math symbols should be named as `\math` followed by the glyph name, for example `\mathonequarter` or `\mathsterling`. Commands which can be used in text or math can then be defined using `\ifmmode`, for example: New description 1994/12/01

```
\DeclareRobustCommand{\pounds}{%
   \ifmmode \mathsterling \else \textsterling \fi
}
```

Note that commands defined in this way must be robust, in case they get put into a section title or other moving argument.

6.5 The order of declaration

NFSS forces you to give all declarations in a specific order so that it can check whether you have specified all necessary information. If you declare objects in the wrong order, it will complain. Here are the dependencies that you have to obey: New description 2019/10/15

- `\DeclareFontFamily` checks that the encoding scheme was previously declared with `\DeclareFontEncoding`.
- `\DeclareFontShape` checks that the font family was declared to be available in the requested encoding (`\DeclareFontFamily`).
- `\DeclareSymbolFont` checks that the encoding scheme is valid.
- `\SetSymbolFont` additionally ensures that the requested math version was declared (`\DeclareMathVersion`) and that the requested symbol font was declared (`\DeclareSymbolFont`).
- `\DeclareSymbolFontAlphabet` checks that the command name for the alphabet identifier can be used and that the symbol font was declared.
- `\DeclareMathAlphabet` checks that the chosen command name can be used and that the encoding scheme was declared.
- `\SetMathAlphabet` checks that the alphabet identifier was previously declared with `\DeclareMathAlphabet` or `\DeclareSymbolFontAlphabet` and that the math version and the encoding scheme are known.
- `\DeclareMathSymbol` makes sure that the command name can be used (i.e., is undefined or was previously declared to be a math symbol) and that the symbol font was previously declared.
- When the `\begin{document}` command is reached, NFSS makes some additional checks—for example, verifying that substitution defaults for every encoding scheme point to known font shape group declarations.

6.6 Font series defaults per document family

New feature 2020/02/02

With additional weights and widths being available in many font families nowadays, it is more likely that somebody will want to match, say, a medium weight serif family with a semi-light sans serif family, or that with one family one wants to use the bold-extended face when `\textbf` is used, while with another it should be bold (not extended) or semi-bold, etc. The default values can be altered using the `\DeclareFontSeriesDefault` declaration in packages or document preambles:

`\DeclareFontSeriesDefault [`⟨*meta family*⟩`] {`⟨*meta series*⟩`} {`⟨*series value*⟩`}`

This declaration takes three arguments:

Meta family interface: Can be either `rm`, `sf` or `tt`. This is optional and if not present the next two arguments apply to the overall default.

Meta series interface: Can be `md` or `bf`.

Series value: This is the value that is going to be used when the combination of ⟨*meta family*⟩ and ⟨*meta series*⟩ is requested.

For example,

```
\DeclareFontSeriesDefault[rm]{bf}{sb}
```

would use `sb` (semi-bold) when `\rmfamily\bfseries` is requested in document.

6.7 Handling of current and requested font series and shape

In the original NFSS implementation, the series was a single attribute stored in `\f@series` and so one always had to specify both weight and width together. Hence, it was impossible to typeset a paragraph in a condensed font and inside have a few words in bold weight (but still condensed) without doing this manually by requesting `\fontseries{bc}\selectfont`.

New feature 2020/02/02

The new implementation now works differently by looking both at the current value of `\f@series` and the requested new series and out of that combination selects a resulting series value. Thus, if the current series is `c` and we ask for `b`, we now get `bc`. This is done by consulting a simple lookup table where entries can be added or changed with `\DeclareFontSeriesChangeRule`:

`\DeclareFontSeriesChangeRule {`⟨*current series*⟩`} {`⟨*requested series*⟩`}`
`{`⟨*result*⟩`} {`⟨*alternative result*⟩`}`

The ⟨*current series*⟩ is the value currently stored in `\f@series`, ⟨*requested series*⟩ is the new series requested, ⟨*result*⟩ is the combined value if it exists for the given font family, and ⟨*alternative result*⟩ is a fallback in case ⟨*result*⟩ doesn't exist. The example above now looks like this:

```
\DeclareFontSeriesChangeRule{c}{b}{bc}{}
```

which means: switch to the `bc` series if `c` is current and `b` is (additionally) requested, and if the current font doesn't have the combination, start the normal font substitution, i.e., switch back shape to `n` and if this combination doesn't succeed, switch back series to `m` as well, ending up with `m/n`.

Another example is:

```
\DeclareFontSeriesChangeRule{bc}{sc}{bsc}{bc}
```

which means: if the current series is bold condensed (`bc`) and semi-condensed (`sc`) is requested (additionally), try bold semi-condensed (`bsc`) if available but stay with bold condensed if not.

A special value is `m` which is used to reset both weight and width. In order to reset only one of them, the special values `?m` (reset width) and `m?` (reset weight) are provided, e.g.:

```
\DeclareFontSeriesChangeRule{bc}{m?}{c}{}
```

The corresponding macro `\DeclareFontShapeChangeRule` is also provided for setting database entries for font shapes:

`\DeclareFontShapeChangeRule` {⟨*current shape*⟩} {⟨*requested shape*⟩} {⟨*result*⟩} {⟨*alternative result*⟩}

An example would be:

```
\DeclareFontShapeChangeRule{it}{sc}{scit}{scsl}
```

If italics is the current shape and small caps is requested, switch to `scit` (small caps italics) and if that doesn't exist, try `scsl` (small caps slanted).

Finally, it is also possible to overrule the entries in the lookup tables and forcibly select a series or shape with:

`\fontseriesforce` {⟨*series*⟩} `\fontshapeforce` {⟨*shape*⟩}

With the example above for the `c` series, issuing `\fontseriesforce{b}` means that the series switches to `b` and not to `bc`. Same applies to `\fontshapeforce`.

6.8 Handling of nested emphasis

`\DeclareEmphSequence` {⟨*list of font declarations*⟩}

This declaration takes a comma separated list of font declarations each specifying how increasing levels of emphasis should be handled. For example: New feature 2020/02/02

```
\DeclareEmphSequence{\itshape,%
                     \upshape\scshape,%
                     \itshape}
```

uses italics for the first, small capitals for the second, and italic small capitals for the third level. If there are more nesting levels than provided, declarations stored in `\emreset` (by default `\ulcshape\upshape`) are used for the next level and then the list restarts.

6.9 Providing font family substitutions

`\DeclareFontFamilySubstitution` {⟨*encoding*⟩} {⟨*family*⟩} {⟨*new-family*⟩}

This declaration selects the font family ⟨*new-family*⟩ as replacement for ⟨*family*⟩ in the font encoding ⟨*encoding*⟩. For example, New feature 2020/02/02

```
\DeclareFontFamilySubstitution{LGR}
        {Montserrat-LF}{IBMPlexSans-TLF}
```

tells LaTeX to substitute the sans serif font `Montserrat-LF` in the Greek encoding `LGR` with `IBMPlexSans-TLF` once requested in a document.

7 Additional text symbols – textcomp

New feature 2020/02/02

For a long time the interface to additional text symbols and the text companion encoding TS1 in general was the textcomp package. All the symbols provided by the textcomp package are now available in LaTeX kernel. Furthermore, an intelligent substitution mechanism has been implemented so that glyphs missing in some fonts are automatically substituted with default glyphs that are sans serif if you typeset in \textsf and monospaced if you typeset using \texttt. In the past they were always taken from Computer Modern Roman if substitution was necessary.

This is most noticeable with \oldstylenums which are now taken from TS1 so that you no longer get 1234 but 1234 when typesetting in sans serif fonts and 1234 when using typewriter fonts.

\legacyoldstylenums {⟨*nums*⟩}
\UseLegacyTextSymbols

If there ever is a need to use the original (inferior) definition, then that remains available as \legacyoldstylenums; and to fully revert to the old behavior there is also \UseLegacyTextSymbols. The latter declaration reverts \oldstylenums and also changes the footnote symbols, such as \textdagger, \textparagraph, etc., to pick up their glyphs from the math fonts instead of the current text font (this means they always keep the same shape and do not nicely blend in with the text font).

The following tables show the macros available. The next commands are 'constructed' accents and are built via TeX macros:

\capitalcedilla␣A	Ą	\textcircled␣a	ⓐ
\capitalogonek␣A	Ą		

These accents are available via font encoding. The numbers in third row show the slot number:

\capitalgrave	`	0	\capitalbreve	˘	8
\capitalacute	´	1	\capitalmacron	¯	9
\capitalcircumflex	^	2	\capitaldotaccent	˙	10
\capitaltilde	~	3	\t	͡	26
\capitaldieresis	¨	4	\capitaltie	͡	27
\capitalhungarumlaut	˝	5	\newtie	‿	28
\capitalring	°	6	\capitalnewtie	‿	29
\capitalcaron	ˇ	7			

Table 2 on the next page contains the full list of commands to access the text symbols. Again, the numbers are the slots in the encoding.

The TS1 encoding contains a rich set of symbols which means that several symbols are only available in a few TeX fonts and some, such as the capital accents, not available at all but developed as part of the reference font implementation. In reality, many existing fonts don't provide a full set of glyphs defined in

\textcapitalcompwordmark		23
\textascendercompwordmark		31
\textquotestraightbase	'	13
\textquotestraightdblbase	"	18
\texttwelveudash	‒	21
\textthreequartersemdash	—	22
\textleftarrow	←	24
\textrightarrow	→	25
\textblank	␢	32
\textdollar	$	36
\textquotesingle	'	39
\textasteriskcentered	*	42
\textdblhyphen	=	45
\textfractionsolidus	/	47
\textzerooldstyle	0	48
\textoneoldstyle	1	49
\texttwooldstyle	2	50
\textthreeoldstyle	3	51
\textfouroldstyle	4	52
\textfiveoldstyle	5	53
\textsixoldstyle	6	54
\textsevenoldstyle	7	55
\texteightoldstyle	8	56
\textnineoldstyle	9	57
\textlangle	⟨	60
\textminus	−	61
\textrangle	⟩	62
\textmho	℧	77
\textbigcircle	◯	79
\textohm	Ω	87
\textlbrackdbl	⟦	91
\textrbrackdbl	⟧	93
\textuparrow	↑	94
\textdownarrow	↓	95
\textasciigrave	`	96
\textborn	⋆	98
\textdivorced	⚯	99
\textdied	†	100
\textleaf	🍃	108
\textmarried	⚭	109
\textmusicalnote	♪	110
\texttildelow	~	126
\textdblhyphenchar	=	127
\textasciibreve	˘	128
\textasciicaron	ˇ	129
\textacutedbl	˝	130
\textgravedbl	˵	131
\textdagger	†	132
\textdaggerdbl	‡	133
\textbardbl	‖	134
\textperthousand	‰	135
\textbullet	•	136
\textcelsius	℃	137
\textdollaroldstyle	$	138
\textcentoldstyle	¢	139
\textflorin	f	140
\textcolonmonetary	₡	141
\textwon	₩	142
\textnaira	₦	143
\textguarani	₲	144
\textpeso	₱	145
\textlira	£	146
\textrecipe	℞	147
\textinterrobang	‽	148
\textinterrobangdown	⸘	149
\textdong	₫	150
\texttrademark	™	151
\textpertenthousand	‱	152
\textpilcrow	⁋	153
\textbaht	฿	154
\textnumero	№	155
\textdiscount	⅍	156
\textestimated	℮	157
\textopenbullet	○	158
\textservicemark	℠	159
\textlquill	⁅	160
\textrquill	⁆	161
\textcent	¢	162
\textsterling	£	163
\textcurrency	¤	164
\textyen	¥	165
\textbrokenbar	¦	166
\textsection	§	167
\textasciidieresis	¨	168
\textcopyright	©	169
\textordfeminine	ª	170
\textcopyleft	ↄ	171
\textlnot	¬	172
\textcircledP	℗	173
\textregistered	®	174
\textasciimacron	¯	175
\textdegree	°	176
\textpm	±	177
\texttwosuperior	²	178
\textthreesuperior	³	179
\textasciiacute	´	180
\textmu	µ	181
\textparagraph	¶	182
\textperiodcentered	·	183
\textreferencemark	※	184
\textonesuperior	¹	185
\textordmasculine	º	186
\textsurd	√	187
\textonequarter	¼	188
\textonehalf	½	189
\textthreequarters	¾	190
\texteuro	€	191
\texttimes	×	214
\textdiv	÷	246

Table 2: Text symbols formerly from the textcomp package

\textquotestraightbase	‚	13	\textsection	§	167
\textquotestraightdblbase	„	18	\textcopyright	©	169
\textcapitalcompwordmark		23	\textordfeminine	ª	170
\textascendercompwordmark		31	\textlnot	¬	172
\textdollar	$	36	\textregistered	®	174
\textquotesingle	'	39	\textdegree	°	176
\textasteriskcentered	*	42	\textpm	±	177
\textdagger	†	132	\textparagraph	¶	182
\textdaggerdbl	‡	133	\textperiodcentered	·	183
\textperthousand	‰	135	\textordmasculine	º	186
\textbullet	•	136	\textonequarter	¼	188
\texttrademark	™	151	\textonehalf	½	189
\textcent	¢	162	\textthreequarters	¾	190
\textsterling	£	163	\texttimes	×	214
\textyen	¥	165	\textdiv	÷	246
\textbrokenbar	¦	166			

Table 3: Symbols available in all TS1 sub-encodings

\textcircled	◯	acc

Table 4: Symbol unavailable in TS1 sub-encoding 1 and higher

TS1 encoding and the question arises: "Which glyphs of the TS1 encoding are implemented by which font?"

Fonts can be ordered in sub-encodings with the \DeclareEncodingSubset macro: New feature 2021/06/01

\DeclareEncodingSubset {⟨*encoding*⟩} {⟨*font family*⟩} {⟨*subset number*⟩}

The macro takes 3 mandatory arguments: An ⟨*encoding*⟩ for which a subsetting is wanted (currently only TS1), the ⟨*font family*⟩ for which we declare the subset and finally the ⟨*subset number*⟩ between 0 (all of the encoding is supported) and 9 (many glyphs are missing). Hence, it is assumed that some symbols are always available by all fonts and each sub-encoding defines macros which become unavailable (i.e., they are not provided in the sub-encoding with that number and all sub-encodings with higher numbers.)

Thus, the symbols that are available in sub-encoding x are the symbols in table 3 (always available) and the symbols that only become unavailable in sub-encodings $> x$. The tables 4 to 12 on pages 36–38 show the symbols that become unavailable in the different sub-encodings. Again, the numbers are the slots in the TS1 encoding, acc indicates a 'constructed' accent.

As an example, \DeclareEncodingSubset{TS1}{foo}{5} indicates that the font family foo contains the always available symbols (table 3) and the ones disabled in sub-encodings 6–9, i.e., tables 9 to 12 on pages 37–38.

As these days many font families are set up to end in -LF (lining figures), -OsF (oldstyle figures), etc. the declaration supports a shortcut: if the ⟨*font family*⟩ name ends in -* then the star gets replaced by these common ending, e.g.,

```
\DeclareEncodingSubset{TS1}{Alegreya-*}{2}
```

\capitalcedilla	¸	acc	\textdivorced	o|o	99
\capitalogonek	˛	acc	\textdied	†	100
\capitalgrave	`	0	\textleaf	🍃	108
\capitalacute	´	1	\textmarried	⚭	109
\capitalcircumflex	^	2	\textmusicalnote	♪	110
\capitaltilde	~	3	\texttildelow	~	126
\capitaldieresis	¨	4	\textdblhyphenchar	=	127
\capitalhungarumlaut	˝	5	\textasciibreve	˘	128
\capitalring	°	6	\textasciicaron	ˇ	129
\capitalcaron	ˇ	7	\textacutedbl	″	130
\capitalbreve	˘	8	\textgravedbl	‶	131
\capitalmacron	¯	9	\textdollaroldstyle	$	138
\capitaldotaccent	˙	10	\textcentoldstyle	¢	139
\capitaltie	‿	27	\textnaira	₦	143
\newtie	‿	28	\textguarani	₲	144
\capitalnewtie{}	‿	29	\textpeso	₱	145
\textdblhyphen	=	45	\textrecipe	℞	147
\textzerooldstyle	0	48	\textpertenthousand	‱	152
\textoneoldstyle	1	49	\textpilcrow	¶	153
\texttwooldstyle	2	50	\textbaht	฿	154
\textthreeoldstyle	3	51	\textdiscount	⅋	156
\textfouroldstyle	4	52	\textopenbullet	○	158
\textfiveoldstyle	5	53	\textservicemark	℠	159
\textsixoldstyle	6	54	\textlquill	{	160
\textsevenoldstyle	7	55	\textrquill	}	161
\texteightoldstyle	8	56	\textasciidieresis	¨	168
\textnineoldstyle	9	57	\textcopyleft	ↄ	171
\textmho	℧	77	\textcircledP	Ⓟ	173
\textbigcircle	◯	79	\textasciimacron	¯	175
\textlbrackdbl	⟦	91	\textasciiacute	´	180
\textrbrackdbl	⟧	93	\textreferencemark	※	184
\textasciigrave	`	96	\textsurd	√	187
\textborn	⋆	98			

Table 5: Symbols unavailable in TS1 sub-encoding 2 and higher

\textlangle	⟨	60	\textrangle	⟩	62

Table 6: Symbols unavailable in TS1 sub-encoding 3 and higher

\textleftarrow	←	24	\textcolonmonetary	₡	141
\textrightarrow	→	25	\textwon	₩	142
\textuparrow	↑	94	\textlira	£	146
\textdownarrow	↓	95	\textdong	₫	150

Table 7: Symbols unavailable in TS1 sub-encoding 4 and higher

\textnumero	№	155	\textestimated	℮	157

Table 8: Symbols unavailable in TS1 sub-encoding 5 and higher

\textflorin	f	140	\textcurrency	¤	164

Table 9: Symbols unavailable in TS1 sub-encoding 6 and higher

\textfractionsolidus	/	47	\textohm	Ω	87
\textminus	−	61	\textmu	µ	181

Table 10: Symbols unavailable in TS1 sub-encoding 7 and higher

\textblank	ƀ	32	\textinterrobangdown	¡	149
\textinterrobang	‽	148			

Table 11: Symbols unavailable in TS1 sub-encoding 8 and higher

\texttwelveudash	—	21	\texttwosuperior	²	178
\textthreequartersemdash	—	22	\textthreesuperior	³	179
\textbardbl	‖	134	\textonesuperior	¹	185
\textcelsius	℃	137			

Table 12: Symbols unavailable in TS1 sub-encoding 9

is the same as writing

```
\DeclareEncodingSubset{TS1}{Alegreya-LF}  {2}
\DeclareEncodingSubset{TS1}{Alegreya-OsF} {2}
\DeclareEncodingSubset{TS1}{Alegreya-TLF} {2}
\DeclareEncodingSubset{TS1}{Alegreya-TOsF}{2}
```

If only some are needed then one can define them individually but in many cases all four are wanted, hence the shortcut.

Maintainers of font bundles that include TS1 encoded font files should add an appropriate declaration into the corresponding ts1*family*.fd file, because otherwise the default subencoding is assumed, which is probably disabling too many glyphs that are actually available in the font.[3]

8 If you need to know more ...

The tracefnt package provides for tracing the actions concerned with loading, substituting and using fonts. The package accepts the following options:

New description 1996/06/01

errorshow Write all information about font changes, etc. but only to the transcript file unless an error occurs. This means that information about font substitution will not be shown on the terminal.

warningshow Show all font warnings on the terminal. This setting corresponds to the default behavior when this tracefnt package is *not* used!

infoshow Show all font warnings and all font info messages (that are normally only written to the transcript file) also on the terminal. This is the default when this tracefnt package is loaded.

debugshow In addition to what is shown by infoshow, show also changes of math fonts (as far as possible): beware, this option can produce a large amount of output.

[3] The LaTeX format contains declarations for many font families already, but this is really the wrong place for the declarations. Thus for new fonts they should be placed into the corresponding .fd file.

loading Show the names of external font files when they are loaded. This option shows only 'newly loaded' fonts, not those already preloaded in the format or the class file before this `tracefnt` package becomes active.

pausing Turn all font warnings into errors so that LaTeX will stop.

Warning: The actions of this package can change the layout of a document and even, in rare cases, produce clearly wrong output, so it should not be used in the final formatting of 'real documents'.

References

[1] Frank Mittelbach and Michel Goossens. *The LaTeX Companion second edition.* With Johannes Braams, David Carlisle, and Chris Rowley. Addison-Wesley, Reading, Massachusetts, 2004.

[2] Donald E. Knuth. Typesetting concrete mathematics. *TUGboat*, 10(1):31–36, April 1989.

[3] Leslie Lamport. *LaTeX: A Document Preparation System.* Addison-Wesley, Reading, Massachusetts, second edition, 1994.