cfr-lm

Clea F. Rees*

v1.8a (SVN Rev: 10794) 2025/02/11

Abstract

cfr-lm offers enhanced support for Latin Modern 2.004 in TEX/LATEX on 8-bit engines[1]. A number of features of the Latin Modern fonts are not easily accessible via the default TEX/LATEX support provided in the official distribution. This package aims to provide TEX/LATEX support for a number of these features including various styles of digits, upright italic and oblique small-caps shapes, and alternative weights and widths. It also supports the variable width typewriter, 'dunhill' and 'quotation' fonts. If microtype is loaded, the package ensures the custom settings designed for Computer and Latin Modern are loaded.

Contents

*Bug tracker: codeberg.org/cfr/nfssext/issues | Code: codeberg.org/cfr/nfssext | Mirror: github.com/cfr42/nfssext

[1]It is not required, does not support and should not be loaded with Unicode engines. Indeed, it was largely written because I wanted to use features easily accessible on Unicode engines while continuing to compile with pdfTEX, which was significantly more reliable at the time and is still considerably faster.

1 Introduction

This document explains how to use the cfr-lm package to access advanced features of the Latin Modern fonts not otherwise supported by the official lm distribution. These features include various styles of digits, upright italic and oblique small-caps italic, alternative weights and widths, and Latin Modern Mono Prop (variable width typewriter), Dunhill and Sans Quotation. By default, the LATEX package provided by `cfr-lm.sty` uses proportional oldstyle digits and variable width typewriter but this can be changed by passing appropriate options when loading the package. The package also supports using e.g. different styles of digits within a document so it is possible to use proportional oldstyle digits by default, say, but tabular lining digits within a particular table. Finally, a command to access the zeroslash character is provided.

cfr-lm version 1.3 requires version 2.004 of GUST's Latin Modern fonts, including the support package provided for TEX. The fonts and TEX support are included in many TEX distributions or may be obtained from http://www.gust.org.pl/projects/e-foundry/latin-modern or your nearest CTAN mirror.

cfr-lm consists of all files listed in manifest.txt and these files are released under the LATEX Project Public Licence as explained in the included licensing notices.

Version 1.3 of the package benefited greatly from feedback provided by Enrico Gregorio, who essentially rewrote the style file using keyval to show me how I ought to be setting the various options up, and Lars Hellström who demonstrated considerable patience in answering my many questions about using fontinst and some peculiarities of the Latin Modern fonts. I hope the changes in the production of the virtual fonts will improve accent placement in 'faked' glyphs (i.e. in the case of characters not included in the EC/T1 font encoding which TEX therefore creates by combining glyphs which are included). The changes involve ignoring all font dimensions given in the AFM files and taking them from the relevant TFM files supplied with Latin Modern instead. The exception to this is the value of `acccapheight` which is set to zero in the TFMs. The current virtual font setup uses fontinst's default value in this case.

If you load microtype, version 1.4 and later will automatically figure out the family-specific settings to use. This is done using aliases which tell microtype to treat the virtual fonts provided by this package in the same way it treats Latin Modern and Computer Modern Roman. See section 7 for details.

2 Requirements

In addition to the usual suspects (LaTeX etc.), the LaTeX support provided by `cfr-lm.sty` requires:

- lm: Latin Modern version 2.004[2]
- nfssext-cfr

If you wish to compile (as opposed to read) the package documentation, additional, packages are required. See `cfr-lm.dtx` for details.

If you wish to recreate the font support files from the base 'lm' package, the easiest option is to download the source from CODEBERG or GITHUB. You can, however, also recreate the font files by hand using the sources included in the CTAN archive alone[3].

3 Limitations

Unlike the official TeX support for Latin Modern, cfr-lm supports only the EC/T1 and Text Companion (TS1) encodings for text. Also unlike the official support, the EC/T1 support depends entirely on virtual fonts. If virtual fonts have disadvantages, then, whatever those disadvantages may be, cfr-lm will inherit them. This does not apply to the TS1 encoding or to mathematics, since these rely purely on the support provided by the official distribution so should be identical.

LaTeX does not recognise the fonts provided by this package, including the TS1 encoding but excluding the mathematics, *as* Latin Modern. This is problematic because newer kernels treat Computer Modern and Latin Modern differently, but only if they are accessed using the default names. This causes at least two complications.

First, the kernel responds to cfr-lm far more 'noisily' than one would like, especially since the noise is entirely unnecessary. The warnings occur because LaTeX switches the default bold series from bx to b unless the document fonts are on a list which includes Latin Modern only by the names provided by lmodern. As far as I can tell, the 'noise' is merely annoying: the actual fonts used and the final output are unaffected, since the kernel tries bx if b is unavailable, though there is presumably some impact on compilation time. In any case, the new version of nfssext-cfr now patches the code the kernel executes at the beginning of the `document` environment by simply adding the appropriate names to the list of Computer and Latin Modern families.

Second, the virtual fonts provided by this package aren't recognised as supporting the `ts1` encoding, so cfr-lm needs to specify this specifically on newer kernels[4].

4 Font setup

As explained above, the fonts use the EC/T1 and Text Companion (TS1) encodings. The provision for the TS1 and mathematics encodings simply calls the support provided by lm. The cfr-lm support simply ensures that access is provided automatically when the T1-encoded virtual fonts it provides are active.

[2] This package should not be used with any other version of Latin Modern due to likely changes to the font metrics, glyph names etc.

[3] The font definition files will be functionally equivalent to those included in the package, but the spacing in some `\DeclareFontFamily` lines will differ because fontinst doesn't write arguments passed to `\installfamily` verbatim to the output stream.. Other files should be equivalent modulo commented lines.

[4] On older kernels, the package continues to load textcomp as it did before.

Table 1: Font families

LM Names	Family	Digits/figures	Notes
Latin Modern Roman	clm	tabular, lining	similar to lm `rm` default
	clm2	proportional, lining	
	clmj	tabular, oldstyle	
	clm2j	proportional, oldstyle	cfr-lm `rm` default
Latin Modern Sans	clms	tabular, lining	similar to lm `sf` default
	clm2s	proportional, lining	
	clmjs	tabular, oldstyle	
	clm2js	proportional, oldstyle	cfr-lm `sf` default
Latin Modern Mono[a]	clmt, clm2t	tabular, lining	similar to lm `tt` default
	clmjt, clm2jt	tabular, oldstyle	
Latin Modern Mono Prop[b]	clmv	tabular, lining	
	clm2v	proportional, lining	
	clmjv	tabular, oldstyle	
	clm2jv	proportional, oldstyle	cfr-lm `tt` default
Latin Modern Sans Quotation	clmqs	tabular, lining	
	clm2qs	proportional, lining	
	clmjqs	tabular, oldstyle	
	clm2jqs	proportional, oldstyle	
Latin Modern Roman Dunhill	clmd	tabular, lining	
	clm2d	proportional, lining	
	clmdj	tabular, oldstyle	
	clm2dj	proportional, oldstyle	

[a] The duplication in TEX name here is to avoid TEX complaining if commands to use proportional digits are issued while one of these fonts is active and to ensure that it is possible to switch smoothly to these fonts if another font with proportional digits is active.
[b] Despite the apparent contradiction in their name, this is variable-width typewriter.

4.1 Font families

Table 1 list the font families provided for use in the EC/T1 and Text Companion (TS1) encodings.

4.2 Shapes, Weights and Widths

Shape, eight and width availability is shown in table 2.

Where applicable, oblique small-caps are substituted for italic small-caps; italic or oblique for upright italic; oblique for italic; and upright for small-caps. This means that some of the commands described in section 6 will fail silently to avoid undue clutter in the log file.

5 The LaTeX package

To load this package, write `\usepackage{cfr-lm}` in your document preamble. By default, the package will define clm2j, clm2js and clm2jv as the default roman/serif, sans and typewriter fonts but you can control the choice by passing options to the package.

The package recognises four keys summarised in table 3 and detailed below. Three of these keys take various options which take the value true or false. These control the default style of figures to be used for each of roman/serif, sans and typewriter text, and whether variable or monowidth typewriter will be used by default.

Table 2: Shapes, weights & widths

family	widths	weights	shapes
clm, clm2, clmj, clm2j	standard	normal	upright, oblique, italic, upright italic, small-caps, oblique small-caps
		bold	upright, oblique, italic
		demi	upright, oblique
clms, clm2s, clmjs, clm2js	standard	normal	upright, oblique
		bold	upright, oblique
	condensed	demi	upright, oblique
clmt, clm2t, clmjt, clm2jt	standard	normal	upright, oblique, italic, small-caps, oblique small-caps
		bold	upright, oblique
		light	upright, oblique
	condensed	light	upright, oblique
clmv, clm2v, clmjv, clm2jv	standard	normal	upright, oblique
		bold	upright, oblique
		light	upright, oblique
clmqs, clm2qs, clmjqs, clm2jqs		normal	upright, oblique
		bold	upright, oblique
clmd, clm2d, clmdj, clm2dj	standard	normal	upright, oblique
		bold	upright, oblique

Table 3: Package options

key	affects	option	possible values
rm	oldstyle/osf[a] lining/lf[a] proportional/prop tabular/tab	true, false	default roman/serif figure style
sf	oldstyle/osf[a] lining/lf[a] proportional/prop tabular/tab	true, false	default sans figure style
tt	oldstyle/osf[a] lining/lf[a] proportional/prop tabular/tab	true, false	default typewriter figure style
	monowidth/mono variable/var	true, false	default typewriter family
qt	—	true, false	nothing unless \qtfont is defined

[a] Lining figures have zero depth i.e. they stand with their bottoms on the current baseline. Oldstyle ('hanging') figures may have depth as well as height i.e. they sit on the baseline with their bottoms hanging over the edge. These options are mutually exclusive and exhaustive.

[b] Proportional figures have variable widths, depending on the widths of the digits e.g. '1' is typically narrower than '6'. Tabular figures have standard, constant width i.e. '1' is as wide as '6', so there is typically more space on each side of '1' than '6'. These options are mutually exclusive and exhaustive.

`rm` (*opt.*) = ⟨*key-value list*⟩

Sets the default style of figures for roman (serif).

`rm/oldstyle` (*opt.*) = `true|false`
`rm/osf` (*opt.*)

Whether to use oldstyle/hanging figures by default.

`rm/lining` (*opt.*) = `true|false`
`rm/lf` (*opt.*)

Whether to use lining figures by default.

Note that `oldstyle` and `osf` are equivalent, while `lining` or `lf` sets the same option but inverted. That is, the following are equivalent:

```
rm={lining=true}
rm={lining}
rm={oldstyle=false}
rm={osf=false}
```

`rm/proportional` (*opt.*) = `true|false`
`rm/prop` (*opt.*)

Whether to use proportional figures by default.

`rm/tabular` (*opt.*) = `true|false`
`rm/tab` (*opt.*)

Whether to use tabular figures by default.

Note that `proportional` and `prop` are equivalent, while `tabular` or `tab` set the same option but inverted. That is, the following are equivalent:

```
rm={tabular=true}
rm={tabular}
rm={tab=true}
rm={tab}
rm={proportional=false}
rm={prop=false}
```

`sf` (*opt.*) = ⟨*key-value list*⟩

Set default figure style for sans serif.

The available keys and values are identical to those for serif explained above.

`sf/oldstyle` (*opt.*) = `true|false`
`sf/osf` (*opt.*)

Whether to use oldstyle/hanging figures by default.

`sf/lining` (*opt.*) = `true|false`
`sf/lf` (*opt.*)

Whether to use lining figures by default.

`sf/proportional` (*opt.*) = `true|false`
`sf/prop` (*opt.*)

Whether to use proportional figures by default.

`sf/tabular` (*opt.*) = `true|false`
`sf/tab` (*opt.*)

Whether to use tabular figures by default.

The available keys and values are identical to those for serif explained above.

`tt` (*opt.*) = ⟨*key-value list*⟩

Set defaults for typewriter. These determine not only the default figure style, but also the default style of other characters.

The available keys and values for setting the default figure style are identical to those for serif explained above. The additional keys for choosing between variable- and mono-width typewriter are explained below.

`tt/oldstyle` (*opt.*) = `true|false`
`tt/osf` (*opt.*)
Whether to use oldstyle/hanging figures by default.

`tt/lining` (*opt.*) = `true|false`
`tt/lf` (*opt.*)
Whether to use lining figures by default.

`tt/proportional` (*opt.*) = `true|false`
`tt/prop` (*opt.*)
Whether to use proportional figures by default.

`tt/tabular` (*opt.*) = `true|false`
`tt/tab` (*opt.*)
Whether to use tabular figures by default.

`tt/monowidth` (*opt.*) = `true|false`
`tt/mono` (*opt.*)
Whether to use mono-width typewriter by default.

`tt/variable` (*opt.*) = `true|false`
`tt/var` (*opt.*)
Whether to use variable-width typewriter by default.

Note that `variable` and `var` are equivalent, while `monowidth` or `mono` set the same option but inverted. That is, the following are equivalent:

```
tt={monowidth=true}
tt={monowidth}
tt={mono=true}
tt={mono}
tt={variable=false}
tt={var=false}
```

`qt` (*opt.*) = `true|false` The fourth key itself takes a true or false value but has no effect unless `\qtfont` is already defined[5].

The default value in all cases is `true` if an option is given without a value. For example, `rm={oldstyle=true}` is equivalent to `rm={oldstyle}`. Many of the options are provided for ease of use but are essentially equivalent. For example, `proportional=false` is equivalent to `tabular=true`. This means that the following two commands are equivalent:

```
\usepackage[%
  rm={lining=true,tabular=false},%
  sf={oldstyle,proportional},%
  tt={oldstyle=false,proportional=true,monowidth}%
]{cfr-lm}

\usepackage[%
  rm={oldstyle=false, proportional=true},%
  sf={lining=false,tabular=false},%
  tt={lining,proportional,variable=false}%
]{cfr-lm}
```

Loading the package without options is equivalent to:

```
\usepackage[%
  rm={oldstyle=true,proportional=true},%
  sf={oldstyle=true,proportional=true},%
  tt={oldstyle=true,proportional=true,variable=true},%
```

[5] This key is designed to control use of LM Sans Quotation in conjunction with prior redefinitions of appropriate environments. Since this is not the sort of redefining a font package should be doing, the option will have absolutely no effect unless you do some prior work to make use of it. In any case, the font can still be accessed directly using the commands explained in section 6.

Table 4: Width macros

width	width command	text command
standard	\regwidth	\textrw{}
condensed	\cdwidth	\textcd{}

```
  qt=false%
]{cfr-lm}
```

That is, by default, oldstyle, proportional figures for roman, sans and typewriter text and variable width typewriter will be selected.

6 Additional font commands

cfr=lm loads nfssext-cfr which is an extension of the package nfssext supplied by Philipp Lehman as part of The Font Installation Guide. The file extends the font selection commands to facilitate access to various font features. Both the original and the extension are designed for use with a wide range of fonts. For this reason, only a subset of the additional commands are relevant to any particular font support package. Those relevant to cfr-lm are described below.

6.1 nfssext-cfr

These commands are available when cfr-lm is loaded. If for some reason you wish to make them available at any other time, use `\usepackage{nfssext-cfr}` in your document preamble.

Note that only combinations supported by the fonts will appear as expected because the commands will only have an effect if the active font offers the relevant variant. For example, trying to switch to a condensed width will have no effect if any of the LM Roman fonts is active. This means that only a subset of combinations are possible. In other cases, one of two things should happen. First, a 'silent' substitution may be made. For example, if you request proportional figures while using monowidth typewriter, tabular figures will be silently substituted. Second, console messages may warn you that the combination you tried to use isn't available. If you request titling while using monowidth typewriter, a console message will warn you it was unavailable. The file `clm-test.tex` gives an idea of what's possible and also serves as an example illustrating some of the commands provided by cfr-lm and other ways of accessing the fonts..

6.1.1 Widths

`\regwidth` `\textrw` `\cdwidth` `\textcd`

Additional macros for changing width are listed in table 4. To switch to an condensed width until further notice, for example, you could use `\cdwidth`. Or use `\texttm{\textlg{\textcd{Hello, world!}}}` to typeset just the text `Hello, world!` in light-weight condensed monowidth typewriter.

Note that the easiest way to switch to semi-bold condensed sans is to resort to using `\fontseries` directly.

```
\textsf{\fontseries{sbc}\selectfont Semi-bold condensed sans}
```

produces

Semi-bold condensed sans

Table 5: Weight macros

weight	weight command	text command
light	\lgweight	\textlg{}
semi-bold	\sbweight	\textsb{}

The problem with using the commands provided by nfssext-cfr is that they are designed, like standard commands such as \scshape, to change *one* aspect of the font at a time[6]. Issuing \textsf{\textcd{\textsb{}}} and \textsf{\textsb{\textcd{}}} are equivalent to \textsf{} because neither standard-width semi-bold nor condensed normal-weight sans is available. The problem is that each command is processed independently, so both switches fail.

Similar considerations in the case of light condensed monowidth typewriter mean that the *order* in which commands are issued is critical. In this case, a light-weight standard-width font is available, but no normal-weight condensed font is provided. Consequently,

```
\texttm{\textlg{\textcd{a successful switch}}}
```

will produce a successful switch while

```
\texttm{\textcd{\textlg{an unsuccessful switch}}}
```

will result in an unsuccessful switch and a warning in the log. In this case, the latter command is equivalent to \texttm{\textlg{}} because \textcd{} can only succeed *after* \textlg{}.

6.1.2 Weights

Additional macros for changing the font weight are given in table 5.

```
\textsb{Semi-bold and \textsl{semi-bold oblique} serif}\\
\texttt{\textlg{Light typewriter}}
```

produces:

Semi-bold and *semi-bold oblique* serif
Light typewriter

6.1.3 Shapes

\sishape
\textsi
\uishape
\textui

Extended shape-changing macros are listed in table 6.

```
\textsc{\textsl{I \emph{always} avoid a kangaroo.}}\\
\textsc{\textit{I \emph{always} avoid a kangaroo.}}\\
\textsl{\textsc{I \emph{always} avoid a kangaroo.}}\\
\textit{\textsc{I \emph{always} avoid a kangaroo.}}\\
\textsi{I \emph{always} avoid a kangaroo.}\\
\textui{Nobody is despised who can manage a crocodile.}
```

produces:

[6] This is, of course, by design. The problem with using standard commands such as \bfseries is that they are designed to change *two* aspects of the font at a time i.e. width and weight.

Table 6: Shape macros

shape	shape command	text command
oblique small-caps	`\slshape\scshape`[a]	`\textsl{\textsc{}}`[a]
	`\scshape\slshape`[a]	`\textsc{\textsl{}}`[a]
	`\itshape\scshape`[b]	`\textit{\textsc{}}`[b]
	`\scshape\itshape`[b]	`\textsc{\textit{}}`[b]
	`\sishape`[b]	`\textsi{}`[b]
upright italic	`\uishape`	`\textui{}`

[a] Supported for all versions of LaTeX 2ε.
[b] Actually the command switches to *italic* small-caps but since LM does not offer this, oblique small-caps are substituted. Unlike their upper/lower-case cousins, small-caps generally look the same whether the font designer calls them 'italic small-caps' or 'pblique small-caps', so the substitution is in no way second-best in this case.

I ALWAYS *AVOID A KANGAROO.*
I ALWAYS *AVOID A KANGAROO.*
I ALWAYS *AVOID A KANGAROO.*
I ALWAYS *AVOID A KANGAROO.*
I ALWAYS *AVOID A KANGAROO.*
Nobody is despised who can manage a crocodile.

if oblique small-caps/upright italic is available for the active font. If it is not, another shape will be substituted.

```
\textsf{\textsc{\slshape The bit about the kangaroo was from Lewis Carroll.}}\\
\textbf{\textui{Sylvia snorkeled snappily.}}
```

produces only:

The bit about the kangaroo was from Lewis Carroll.
Sylvia snorkeled snappily.

where upright sans and bold italic are substituted for italic small-caps sans and bold upright italic since neither is available. Note that the first substitution produces a warning in the log while the second is done 'silently'.

6.1.4 Figures

`\lstyle` `\ostyle` `\pstyle` `\tstyle` Commands are provided to change either one or both aspects of digits' style (table 7. The macros all use standard abbreviations and have predictable forms. Any macro of the form `\⟨θ⟩style` switches a single aspect of the current digits' style. ⟨θ⟩ may be one of `l`, `o`, `p` or `t` which represent *l*ining, *o*ldstyle, *p*roportional and *tabular* respectively.

`\textl` `\texto` `\textp` `\textt` The corresponding text commands have the format `\text⟨θ⟩`, where ⟨θ⟩ may take the same values as before.

In this document, proportional lining figures are used by default for roman/serif and sans, while tabular lining are used for typewriter:

0123456789
0123456789
0123456789

Table 7: Macros for changing the style of figures.

figure style	style command	text command
lining[a]	\lstyle	\textl{}
oldstyle[b]	\ostyle	\texto{}
proportional[c]	\pstyle	\textp{}
tabular[d]	\tstyle	\textt{}
proportional[c], lining[a]	\plstyle	\textpl{}
proportional[c], oldstyle[b]	\postyle	\textpo{}
tabular[d], lining[a]	\tlstyle	\texttl{}
tabular[d], oldstyle[b]	\tostyle	\textto{}

[a] *l*ining figures stand on the current baseline: 0123456789. They are generally preferable for use in tabulars, mathematics, code listings, diagrams etc. Contemporary usage also favours them in text, even though traditional typography would frown on this.

[b] *o*ldstyle figures sit on or hang from the current baseline: 0123456789. They are generally considered more suitable for use in text than lining figure.

[c] *p*roportional figures take up space in proportion to the actual width of the digit: 0123456789. These are generally preferable in most non-specialised contexts.

[d] *t*abular figures each take up a standard width, regardless of the width of the digit: 0123456789. These are better in tabulars where columns of digits should be aligned, code listings set in monowidth typewriter (as is usual) etc.

but oldstyle figures are also accessible. For example:

```
\texto{0123456789}\\
\textsf{\texto{0123456789}}\\
\texttt{\texto{0123456789}}
```

produces:

0123456789
0123456789
0123456789

First, note that it is necessary to reissue `\texto{}` after switching to sans or typewriter text. This is because both switching to sans or typewriter and switching to another figure style involves a switch of font family[7].

Second, note that the output shows *proportional* oldstyle figures for romand and sans, but *tabular* oldstyle for typewriter, because the command `\texto{}` only changes *one* aspect of the style. Because proportional figures were already active for serif and sans, the command switched to proportional oldstyle figures in the first two cases. Contrariwise, since tabular figures were active for typewriter, the same command switched to tabular oldstyle figures in the third case.

In many cases, it is convenient to switch or ensure both aspects of digits together e.g. to ensure tabular lining figures are used in tabulars. Four macros are provided for this purpose. These have the form `\⟨θ⟩⟨β⟩style`, where ⟨θ⟩ may be either `p` or `t` and ⟨β⟩ may be either `l` or `o`.

The corresponding text commands have the format `\text⟨θ⟩⟨β⟩`.

Taking roman as an example, tabular oldstyle digits may be accessed in several ways:

```
\texto{\textt{0123456789}}\\
\textt{\texto{0123456789}}\\
```

\plstyle \postyle \tlstyle \tostyle \textpl \textpo \texttl \textto

[7] Compare a switch in width or weight which does not typically involve a change of active font family.

Table 8: Macros for switching to mono-/variable-width typewriter

typewriter font	style command	text command
variable typewriter	\tvstyle	\texttv{}
monowidth typewriter	\tmstyle	\texttm{}

Table 9: LM Sans Quotation

sans quotation	\qtstyle	\textqt{}

```
\textto{0123456789}
```

will produce three identical lines of figures:

0123456789
0123456789
0123456789

6.1.5 Typewriter variants

\texttv
\tvstyle
\texttm
\tmstyle

In addition to the package options to specify either LM Mono or LM Mono Prop as default (i.e. either monowidth or variable-width typewriter), it is possible to access the non-default font using the commands in table 8.

Mono-width is default in this document so

```
\texttt{This is monowidth width typewriter.}\\
\texttv{This is variable typewriter} \texttm{except this bit at the end.}
```

produces:

This is monowidth width typewriter.
This is variable typewriter except this bit at the end.

6.1.6 Latin Modern Sans Quotation

\textqt
\qtstyle

Latin Modern Sans Quotation can be accessed using the macros listed in table 9.

For example, `\textqt{some text in the font}` will produce some text in the font.

6.1.7 Latin Modern Roman Dunhill

\textti
\tistyle

Latin Modern Roman Dunhill can be accessed using the macros listed in table 10.

To ensure the command succeeds independently of the currently active font, you may wish to issue `\normalfont` first. For example:

```
\normalfont\textti{Kinky Querulous Rhinos X-Ray Exultant Risque Zebras}\\
```

Table 10: Latin Modern Roman Dunhill

style	style command	text command
titling	\tistyle	\textti{}

```
\textti{\textsl{Kinky Querulous Rhinos X-Ray Exultant Risque Zebras}}
```

produces:

Kinky Querulous Rhinos X-Ray Exultant Risque Zebras
Kinky Querulous Rhinos X-Ray Exultant Risque Zebras

6.2 zeroslash

`\zeroslash` cfr-lm provides one additional command. `\zeroslash` will produce the 0 character from the current font.

7 Microtype

Support for family-specific microtypographical features supported by microtype. This code will do nothing if you do not load microtype. If you do use these features, all regular roman and sans families, together with the sans quotation font, will use the settings for Computer Modern Roman. The fallback generic settings will continue to be applied to the typewriter and 'dunhill' families.

A Installation

The vast majority of users should IGNORE this section entirely. cfr-lm is included in all major TEX distributions and should be installed as part of your TEX installation. Installing the package yourself should be done only as a last resort or an educational exercise.

Note, in particular, that this version of cfr-lm should **not** be installed on older LATEX kernels as it is designed to work with the (New) New Font Selection Scheme, as updated around 2020[8]. Use the initial release of cfr-lm if your installation of LATEX predates those changes.

Installation varies with TEX distribution so you should consult the documentation which came with your system for details. In most cases, you will need to perform three steps:

1. move or copy the package files to appropriate locations on your system;
2. refresh the TEX database;
3. incorporate the included map file fragments for the different engines your distribution supports.

The following instructions assume you are using TEX Live[9]. They should not be too difficult to adapt if you are using a different distribution.

A.1 Install the files

The files should be installed in one of two locations: *either* the local system-wide TEX tree *or* your personal tree. If the package is installed system-wide, all users will have access to it. On the other hand, you may need privileges you do not have to do this in which case you must use your personal tree.

There are serious disadvantages to installing the package into your personal tree. In particular, these pertain to use of `updmap -user` rather than `updmap -sys`. If you are not aware of these disadvantages, please ensure you are fully cognisant of them before proceeding[10]. Merely removing the package from your personal tree at a later point will *not* undo the effects.

For TEX Live, `kpsewhich -var-value TEXMFLOCAL` will return the path to the local tree and `kpsewhich -var-value TEXMFHOME` the path to your personal tree. The package already includes a hierarchy of files to help you install them correctly. Ignoring any symbolic link in the top directory, move or copy the files in `doc`, `fonts` and `tex` into the appropriate locations. If the tree is initially empty, you can simply move or copy the directories in as they are. If the tree already contains other packages, you may need to merge the package hierarchy with the pre-existing one. For example, if you already have a `doc/fonts` directory, move or copy `doc/fonts/cfr-lm` into `doc/fonts/`. If you have a `doc` directory but not a `doc/fonts`, move `doc/fonts` into `doc/`.

A.2 Refresh the database

Again, this depends on your distribution. For TEX Live, `mktexlsr <path to directory>` for the directory you used in the first step should do the trick. Note that you *may* be able to skip this step if you install into your personal tree. Whether this is so depends on the details of your set-up. As a test, move to a directory containing none of the package files and try `kpsewhich cfr-lm.sty`.

[8] The package should™ work fine on older kernels, but the new version is bound to have some bugs and there is no reason to use it on these systems. The sole purpose of the update is to accommodate the breaking changes made to font selection. If you don't have those changes installed locally, nothing should be broken and the newer version of cfr-lm offers no advantage at all.

[9] This includes MacTEX for OS X users.

[10] See, for example, Why shouldn't I use `getnonfreefonts` to install additional fonts? Why shouldn't I use `updmap` when installing or removing fonts?.

If the file is found, you don't need to refresh the database; otherwise use `mktexlsr` and then try again.

A.3 Install the map fragments

For TEX Live, there are at least two ways of doing this. The second method varies according to the version of TEX Live and instructions are provided accordingly. Both methods depend on whether you installed into `TEXMFLOCAL` or `TEXMFHOME`. If you installed system-wide, the choice is relatively straightforward — it obviously makes sense in that case to update the font maps system-wide as well.

If, on the other hand, you installed into your personal tree, the matter is more complex. On the one hand, updating the system-wide maps may create difficulties or confusion for other users because while the map files will list the fonts as available, they will not be able to access them. On the other hand, maintaining personal font map files can produce difficulties and confusions of its own[11]. Whether it is to be preferred or not is a complex issue and depends on the details of your TEX distribution, local configuration and personal preference. The one clear case is that in which you install into your personal tree because you lack the privileges needed to install system-wide. In that case, you have no choice but to maintain personal font map files or forgo the use of all fonts not provided by your administrator. Other cases are thankfully beyond the scope of this document.

A.3.1 Method 1

If you installed the package system-wide, use the command:

```
updmap-sys --enable Map=clm.map
```

If you installed the package in your personal tree, you *may* prefer[11]:

```
updmap --enable Map=clm.map
```

Either way, `updmap` will output a good deal of information after each incantation. This is normal. Just check that it does not end with an error and that it found the new map file.

A.3.2 Method 2: TEX Live 2008 (and probably earlier)

If you installed the package system-wide, use `updmap-sys --edit`.

If you installed into your personal tree, you *may* prefer to use `updmap --edit`[11].

Either way, a configuration file will be opened which you can edit. Move to the end of the file and add the following line:

```
Map clm.map
```

When you are done, save the file. `updmap` or `updmap-sys` will produce a great deal of output if all is well. Just check that it does not end with an error and that `clm.map` is found.

A.3.3 Method 2: TEX Live 2009 (and possibly later)

If you installed the package system-wide, edit or or create `TEXMFLOCAL/web2c/updmap-local.cfg` and add the following line to the end of the file:

```
Map clm.map
```

[11] See, for example, Why shouldn't I use `getnonfreefonts` to install additional fonts? Why shouldn't I use `updmap` when installing or removing fonts?.

Save the file and tell `tlmgr` to merge in your addition using the command:

```
tlmgr generate updmap
```

`tlmgr` will then tell you that you need to ensure the changes are propagated correctly by calling `updmap-sys`. This should produce a great deal of output. Check that it finds the new map file and does not end with an error.

If you installed into your personal tree, you *may* prefer to use `updmap --edit` as described above for TeX Live 2008[12].

A.3.4 Method 3: Current/Recent TeX Live

If you installed the package system-wide, tell `\updmap` to enable the map file:

```
updmap --sys --enable Map=clm.map
```

This should produce a great deal of output. Check that it finds the new map file and does not end with an error.

If you installed into your personal tree, you *could* use `updmap --user` in place of `updmap --sys` as described above for TeX Live 2008, but this is **not** recommended[12].

To test your installation and that the package works on your system, latex this file (`cfr-lm.tex`). The console output and/or log should tell you whether any fonts were not found. If you are careful not to overwrite it, you may also compare your output with `cfr-lm.pdf`.

B Implementation

You do not need to read the remainder of this document in order to install or use the fonts.

B.1 LaTeX 2ε package

`cfr-lm.sty` (*pkg.*) The LaTeX user interface.

```
\NeedsTeXFormat{LaTeX2e}
\RequirePackage{svn-prov}
\ProvidesPackageSVN[\filebase.sty]{$Id: cfr-lm.dtx 10794 2025-02-11 05:59:23Z cfrees $}[v1.8a
 \revinfo][Extended support for Latin Modern 2.004]
\DefineFileInfoSVN[clm]
\RequirePackage[T1]{fontenc}
\RequirePackage{nfssext-cfr}[2024/01/01]
```

nfssext-cfr provides `\ProcessKeyOptions`, `\IfFormatAtLeastTF` on older kernels and ensures font encoding subset declarations are valid in font definition files if the format is post=2020 but not new enough to support them out-of-the-box. The declarations themselves are inserted by `l3build` using fontscripts's `fntbuild` and defined in `build.lua`[13]. Note that these declarations do **not** conform to the recommendations provided by the LaTeX Project's helper script. Instead, they pick-up the declarations used by the LaTeX Project for Latin Modern. This is done dynamically, so if the developers ever decide to use their script's recommendations for Latin Modern, any changes will be automatically picked up[14].

```
\IfFormatAtLeastTF {2020-02-02}{%
```

[12] See, for example, Why shouldn't I use `getnonfreefonts` to install additional fonts? Why shouldn't I use `updmap` when installing or removing fonts?.

[13] I've been told this shouldn't be on CTAN, so it isn't, but you can find it on GITHUB.

[14] On the other hand, if the implementation details of subset handling in the format change, the code in this package may break. But that's already true for so much of the code in nfssext/nfssext-cfr, that one more area of fragility probably matters but little.

```
  \DeclareEncodingSubset{TS1}{clm}{1}%
  \DeclareEncodingSubset{TS1}{clm2}{1}%
  \DeclareEncodingSubset{TS1}{clm2d}{1}%
  \DeclareEncodingSubset{TS1}{clm2dj}{1}%
  \DeclareEncodingSubset{TS1}{clm2j}{1}%
  \DeclareEncodingSubset{TS1}{clm2jqs}{1}%
  \DeclareEncodingSubset{TS1}{clm2js}{1}%
  \DeclareEncodingSubset{TS1}{clm2jt}{1}%
  \DeclareEncodingSubset{TS1}{clm2jv}{1}%
  \DeclareEncodingSubset{TS1}{clm2qs}{1}%
  \DeclareEncodingSubset{TS1}{clm2s}{1}%
  \DeclareEncodingSubset{TS1}{clm2t}{1}%
  \DeclareEncodingSubset{TS1}{clm2v}{1}%
  \DeclareEncodingSubset{TS1}{clmd}{1}%
  \DeclareEncodingSubset{TS1}{clmdj}{1}%
  \DeclareEncodingSubset{TS1}{clmj}{1}%
  \DeclareEncodingSubset{TS1}{clmjqs}{1}%
  \DeclareEncodingSubset{TS1}{clmjs}{1}%
  \DeclareEncodingSubset{TS1}{clmjt}{1}%
  \DeclareEncodingSubset{TS1}{clmjv}{1}%
  \DeclareEncodingSubset{TS1}{clmqs}{1}%
  \DeclareEncodingSubset{TS1}{clms}{1}%
  \DeclareEncodingSubset{TS1}{clmt}{1}%
  \DeclareEncodingSubset{TS1}{clmv}{1}%
}{%
  \RequirePackage{textcomp}}
\UndeclareTextCommand{\textperthousand}{T1}
\ExplSyntaxOn
```

Parts of this file are based on `lmodern.sty` which is included with the Latin Modern fonts released by GUST and available from http://www.gust.org.pl/projects/e-foundry/latin-modern.

This draws also on the documentation for the microtype package and `MinionPro.sty`. `MinionPro.sty` is available as part of the minionpro package and can be obtain from http://mirror.ctan.org/fonts/minionpro. `MinionPro.sty` is in the public domain. The documentation for microtype is available in English and German from microtype. It is part of the microtype package which is itself licensed under the LPPL.

Since removed?

BEGIN bools

```
\bool_new:N \l__clm_rm_osf_bool
\bool_new:N \l__clm_rm_prop_bool
\bool_new:N \l__clm_sf_osf_bool
\bool_new:N \l__clm_sf_prop_bool
\bool_new:N \l__clm_tt_osf_bool
\bool_new:N \l__clm_tt_prop_bool
\bool_new:N \l__clm_tt_mono_bool
\bool_new:N \l__clm_qt_bool
```

END bools

```
\keys_define:nn { clm }
{
```

`rm/oldstyle` (*opt.*) boolkey for roman osf/lf

```
  rm / oldstyle .bool_set:N = \l__clm_rm_osf_bool,
  rm / oldstyle .default:n = true,
  rm / oldstyle .initial:n = true,
```

`rm/osf` (*opt.*) Shorthand

```
rm / osf .bool_set:N = \l__clm_rm_osf_bool,
rm / osf .default:n = true,
```

`rm/lining` (*opt.*) Inverse

```
rm / lining .bool_set_inverse:N = \l__clm_rm_osf_bool,
rm / lining .default:n = true,
```

`rm/lf` (*opt.*) Shorthand

```
rm / lf .bool_set_inverse:N = \l__clm_rm_osf_bool,
rm / lf .default:n = true,
```

`rm/proportional` (*opt.*) boolkey for roman prop/tab figures

```
rm / proportional .bool_set:N = \l__clm_rm_prop_bool,
rm / proportional .default:n = true,
rm / proportional .initial:n = true,
```

`rm/prop` (*opt.*) Shorthand

```
rm / prop .bool_set:N = \l__clm_rm_prop_bool,
rm / prop .default:n = true,
```

`rm/tabular` (*opt.*) Inverse

```
rm / tabular .bool_set_inverse:N = \l__clm_rm_prop_bool,
rm / tabular .default:n = true,
```

`rm/tab` (*opt.*) Shorthand

```
rm / tab .bool_set_inverse:N = \l__clm_rm_prop_bool,
rm / tab .default:n = true,
```

`sf/oldstyle` (*opt.*) boolkeys for sans osf/lf

```
sf / oldstyle .bool_set:N = \l__clm_sf_osf_bool,
sf / oldstyle .default:n = true,
sf / oldstyle .initial:n = true,
```

`sf/osf` (*opt.*) Shorthand

```
sf / osf .bool_set:N = \l__clm_sf_osf_bool,
sf / osf .default:n = true,
```

`sf/lining` (*opt.*) Inverse

```
sf / lining .bool_set_inverse:N = \l__clm_sf_osf_bool,
sf / lining .default:n = true,
```

`sf/lf` (*opt.*) Shorthand

```
sf / lf .bool_set_inverse:N = \l__clm_sf_osf_bool,
sf / lf .default:n = true,
```

`sf/proportional` (*opt.*) boolkeys for sans prop/tab figures

```
sf / proportional .bool_set:N = \l__clm_sf_prop_bool,
sf / proportional .default:n = true,
sf / proportional .initial:n = true,
```

`sf/prop` (*opt.*) Shorthand

```
sf / prop .bool_set:N = \l__clm_sf_prop_bool,
sf / prop .default:n = true,
```

`sf/tabular` (*opt.*) Inverse

```
sf / tabular .bool_set_inverse:N = \l__clm_sf_prop_bool,
sf / tabular .default:n = true,
```

`sf/tab` (*opt.*) Shorthand

```
sf / tab .bool_set_inverse:N = \l__clm_sf_prop_bool,
sf / tab .default:n = true,
```

`tt/oldstyle` (*opt.*) boolkeys for typewriter osf/lf

```
tt / oldstyle .bool_set:N = \l__clm_tt_osf_bool,
tt / oldstyle .default:n = true,
tt / oldstyle .initial:n = true,
```

`tt/osf` (*opt.*) Shorthand

```
tt / osf .bool_set:N = \l__clm_tt_osf_bool,
tt / osf .default:n = true,
```

`tt/lining` (*opt.*) Inverse

```
tt / lining .bool_set_inverse:N = \l__clm_tt_osf_bool,
tt / lining .default:n = true,
```

`tt/lf` (*opt.*) Shorthand

```
tt / lf .bool_set_inverse:N = \l__clm_tt_osf_bool,
tt / lf .default:n = true,
```

`tt/proportional` (*opt.*) boolkeys for typewriter prop/tab figures

```
tt / proportional .bool_set:N = \l__clm_tt_prop_bool,
tt / proportional .default:n = true,
tt / proportional .initial:n = true,
```

`tt/prop` (*opt.*) Shorthand

```
tt / prop .bool_set:N = \l__clm_tt_prop_bool,
tt / prop .default:n = true,
```

`tt/tabular` (*opt.*) Inverse

```
tt / tabular .bool_set_inverse:N = \l__clm_tt_prop_bool,
tt / tabular .default:n = true,
```

`tt/tab` (*opt.*) Shorthand

```
tt / tab .bool_set_inverse:N = \l__clm_tt_prop_bool,
tt / tab .default:n = true,
```

`tt/monowidth` (*opt.*) boolkeys for typewriter mono/variable width

```
tt / monowidth .bool_set:N = \l__clm_tt_mono_bool,
tt / monowidth .default:n = true,
tt / monowidth .initial:n = false,
```

`tt/mono` (*opt.*) Shorthand

```
  tt / mono .bool_set:N = \l__clm_tt_mono_bool,
  tt / mono .default:n = true,
```

`tt/variable` (*opt.*) Inverse

```
  tt / variable .bool_set_inverse:N = \l__clm_tt_mono_bool,
  tt / variable .default:n = true,
```

`tt/var` (*opt.*) Shorthand

```
  tt / var .bool_set_inverse:N = \l__clm_tt_mono_bool,
  tt / var .default:n = true,
```

`rm` (*opt.*) options for roman

```
  rm .code:n ={
    \keys_set:nn { clm / rm } { #1 }
  },
  rm .value_required:n = true,
  rm .usage:n = load,
```

`sf` (*opt.*) options for sans

```
  sf .code:n ={
    \keys_set:nn { clm / sf } { #1 }
  },
  sf .value_required:n = true,
  sf .usage:n = load,
```

`tt` (*opt.*) options for typewriter

```
  tt .code:n ={
    \keys_set:nn { clm / tt } { #1 }
  },
  tt .value_required:n = true,
  tt .usage:n = load,
```

`qt` (*opt.*) note that this option does nothing unless `\qtfont` is defined appropriately

```
  qt .bool_set:N = \l__clm_qt_bool,
  qt .default:n = true,
  qt .initial:n = false,
  qt .usage:n = load,
}

\providecommand \IfFormatAtLeastTF { \@ifl@t@r \fmtversion }
```

options override defaults

```
\IfFormatAtLeastTF { 2022-06-01 }
{
  \ProcessKeyOptions [ clm ]
}{
  \RequirePackage { l3keys2e }
  \ProcessKeysOptions { clm }
}
\IfFormatAtLeastTF { 2020-10-01 }{
}{
  \RequirePackage { xparse }
  \providecommand \ExpandArgs [1]
```

```
  { \cs_if_exist_use:c { exp_args:N #1 } }
}
```

Translate user/default settings into bits of Berry names.

\rmdefault
\sfdefault
\ttdefault

Make LM default for all families, implementing options for each

```
\tl_gset:Ne \rmdefault
{
  clm \bool_if:NT \l__clm_rm_prop_bool { 2 }
  \bool_if:NT \l__clm_rm_osf_bool { j }
}
\tl_gset:Ne \sfdefault
{
  clm \bool_if:NT \l__clm_sf_prop_bool { 2 }
  \bool_if:NT \l__clm_sf_osf_bool { j } s
}
\tl_gset:Ne \ttdefault
{
  clm \bool_if:NT \l__clm_tt_prop_bool { 2 }
  \bool_if:NT \l__clm_tt_osf_bool { j }
  \bool_if:NTF \l__clm_tt_mono_bool { t } { v }
}
```

\qtfont

Handle the qt option, failing gracefully if somebody has enabled the option without defining `\qtfont` appropriately. We do this in a hook at the start of the document so we can respond if the required macro is defined after cfr-lm is loaded. It would be nice if there was a more satisfactory approach, but I can't think of one a font package has any business implementing.

```
\hook_gput_code:nnn { begindocument } { . }
{
  \bool_if:NT \l__clm_qt_bool
  {
    \cs_if_exist:NTF { \qtfont }
    {
      \qtfont{\qtstyle}
    }{
      \PackageWarning{cfr-lm}
      {
        Option ~ 'qt' ~ cannot ~ be ~ implemented ~ unless \MessageBreak
        '\backslash qtfont' ~ is ~ defined ~ appropriately. ~ This ~ is ~ not \MessageBreak
        done ~ automatically ~ to ~ maximise ~ compatibility ~ with ~ other \MessageBreak
        classes ~ and ~ packages. ~ The ~ suggested ~ use ~ is ~ to ~ have ~ '\backslash
qtfont' \MessageBreak
        redefine ~ a command ~ such ~ as ~ '\backslash quotefont' ~ which ~ is ~ initially
\MessageBreak
        set ~ to ~ some ~ reasonable ~ default ~ and ~ used ~ in ~ the \MessageBreak
        definition ~ of ~ the ~ quote ~ and/or ~ quotation ~ environments. \MessageBreak
        A ~ font ~ package ~ has ~ no ~ business ~ meddling ~ in ~ such \MessageBreak
        things ~ so ~ I'm ~ going ~ to ~ ignore ~ this ~ option
      }
    }
  }
}
%
\ExplSyntaxOff
```

Maths setup is 'based on' (i.e. filched from) lmodern.sty

```
\SetSymbolFont{operators}   {normal}{OT1}{lmr} {m}{n}
\SetSymbolFont{letters}     {normal}{OML}{lmm} {m}{it}
```

```
\SetSymbolFont{symbols}     {normal}{OMS}{lmsy}{m}{n}
\SetSymbolFont{largesymbols}{normal}{OMX}{lmex}{m}{n}
\SetSymbolFont{operators}   {bold}  {OT1}{lmr} {bx}{n}
\SetSymbolFont{letters}     {bold}  {OML}{lmm} {b}{it}
\SetSymbolFont{symbols}     {bold}  {OMS}{lmsy}{b}{n}
\SetSymbolFont{largesymbols}{bold}  {OMX}{lmex}{m}{n}
%
\DeclareFontSubstitution{OT1}{lmr}{m}{n}
\DeclareFontSubstitution{OML}{lmm}{m}{it}
\DeclareFontSubstitution{OMS}{lmsy}{m}{n}
\DeclareFontSubstitution{OMX}{lmex}{m}{n}
%
```

`\mathbf` `\mathsf` `\mathit` `\mathtt` maths alphabets

```
\SetMathAlphabet{\mathbf}{normal}{OT1}{lmr}{bx}{n}
\SetMathAlphabet{\mathsf}{normal}{OT1}{lmss}{m}{n}
\SetMathAlphabet{\mathit}{normal}{OT1}{lmr}{m}{it}
\SetMathAlphabet{\mathtt}{normal}{OT1}{lmtt}{m}{n}
\SetMathAlphabet{\mathbf}{bold}  {OT1}{lmr}{bx}{n}
\SetMathAlphabet{\mathsf}{bold}  {OT1}{lmss}{bx}{n}
\SetMathAlphabet{\mathit}{bold}  {OT1}{lmr}{bx}{it}
\SetMathAlphabet{\mathtt}{bold}  {OT1}{lmtt}{m}{n}
%
```

`\mathsterling`

```
\def\mathsterling{\mathit{\mathchar"70BF}}
```

`\dotdigitenc` `\textdde`

```
\DeclareRobustCommand{\dotdigitenc}{%
  \not@math@alphabet\dotdigitenc\relax
  \fontencoding{U}\selectfont}
\DeclareTextFontCommand{\textdde}{\dotdigitenc}
```

`\zeroslash` The slashed zero.

```
\newcommand*{\zeroslash}{\textdde{\char 250}}
```

Partly from microtype docs; partly from MinionPro package

We need to set up aliases for the font families created by cfr-lm so that microtype recognises them as similar to Computer Modern Roman.

T1 families in cfr-lm: clm clm2 clm2d clm2dj clm2j clm2jqs clm2js clm2jt clm2jv clm2qs clm2s clm2t clm2v clmd clmdj clmj clmjqs clmjs clmjt clmjv clmqs clms clmt clmv

See variants set in nfssext-cfr

Ref: https://tex.stackexchange.com/a/75440

```
\ExplSyntaxOn
```

Aliases for microtype so fonts get the same custom treatment they do with lm.

```
\cs_new_nopar:Nn \__clm_microtype_hook:
{
  \clist_map_inline:nn
  {
    clm,clm2,clm2j,clmj,% roman
    clms,clm2js,clm2s,clmjs,% sans
    clmqs,clm2jqs,clm2qs,clmjqs%
```

```
    %  clm2d,clm2dj,clmd,clmdj,% leave unaliased?
    %  clm2jv,clmjv,clm2v,clmv,% leave unaliased?
    %  clmt,clm2t,clmjt,clm2jt% leave unaliased?
  }{
    \DeclareMicrotypeAlias { ##1 } { cmr }
  }
}
```

If `\Microtype@Hook` exists, add our material to it; if not, create it.

```
\@ifpackageloaded{microtype}
{
  \__clm_microtype_hook:
}{
  \cs_if_free:cTF { Microtype@Hook }
  {% MinionPro has \global before this
    \cs_new_eq:NN \Microtype@Hook \__clm_microtype_hook:
  }{
    \g@addto@macro\Microtype@Hook{\__clm_microtype_hook:}
  }
}
\ExplSyntaxOff

%% end cfr-lm.sty
```

The remaining files are not used directly, but are required to generate the files which allow TEX and LATEX to use the fonts. The sources use `fontinst` and are documented in `cfr-lm-build.dtx` with (sometimes sparse) comments. While you can install these files into a TEX tree, they are not required for typesetting.

Change History

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.