lua-font-config

Clea F. Rees*

v0.3†

Abstract

An opentype font configuration package for LuaLaTeX 2ε. The aim is to provide an alternative to fontspec which is both faster and more powerful, by doing as much processing as possible in Lua. To avoid loading fonts unnecessarily, the data used is corrected during compilation and the results cached for future use.

The package introduces the concept of *font sets* (see section 3.3). The idea is to allow authors to 'switch out' one font configuration for another at any point in the document. Font sets may include specifications for one or more of the standard three text families and/or a Unicode maths font.

The package should be compatible with fontspec (section 4.1), though this is not currently tested. Basic usage is also compatible with unicode-math (section 4.4), but maths font configuration and support for multiple font sets require lua-unicode-math (section 4.3).

This package is experimental.

1 Requirements

The package requires LuaLaTeX 2ε. It will not work with any other engine or format.

Unicode maths and font sets are supported *via* lua-unicode-math. These features will **not** work with unicode-math.

2 Introduction

The key-value user interface and support for font sets are currently highly experimental[1] .

Key-value configuration currently supports setting up the base text font families and Unicode maths fonts (using lua-unicode-math). Multiple independent configurations for maths and text (font sets) may be specified in the preamble, enabling one to switch between them at any point in the document. Overhead is minimal: if a set is never used, no fonts are defined and no families configured. Support for font sets is *doubly experimental and provisional*.

In addition, specialist text families may be configured and accessed *via* customisable command switches (e.g. `\MyFancyTitleFont` or whatever). These may be accessed independently of the current font set.

By default, the package tries to provide font families which are as rich and flexible as possible. For example, many families will include a variety of widths and weights, both italic and oblique upright and small-caps etc. out-of-the-box[2].

*Many thanks to everyone who provided feedback on version 0.1. I'd particularly like to thank Max Chernoff who is primarily responsible for cleaning up the ConTeXt code, made many other suggestions and demonstrated remarkable patience in the face of my ineptitude.

†SVN: 12228 2026/11/24

1 v0.1: 'The user interface is currently **extremely** basic.'
v0.2: 'The user interface is currently currently somewhat basic.'
The lower-level interface is more stable, though still experimental, but this remains *extremely* basic by design.

[2]At present, this depends almost entirely on the quality of the data the fonts provide. It is not always good or, even, vaguely accurate. If things do not fit, some (currently minimal) effort is made to accommodate them as distinct font families, so they are still available at document level.

'Fake' fonts (e.g. bold or oblique) are supported, but the package never adds such effects by default — if no bold or oblique is found, the family will have neither, unless explicitly requested[3].

A reasonable[4] variety of general options and opentype font features now have key-value support for text families. 'Scoped' features, which apply to only some fonts in a family, are also supported.

Advanced users may also pass opentype feature specifications for text families more-or-less directly to the font loader. This enables any loader-supported specification to be used and is faster, but requires knowledge of the syntax and opentype tag names.

Simple usage looks like this:

```
\usepackage{lua-font-config}
\fontconfig{%
  rm = texgyrepagella,
  sf = texgyreheros,
  tt = texgyrecursor,
  math = texgyrepagellamath,
}
```

Or, adding a second set of fonts:

```
\usepackage{lua-font-config}
\fontconfig{%
  rm = texgyrepagella,
  sf = texgyreheros,
  tt = texgyrecursor,
  math = texgyrepagellamath,
}
\newfontset{alts}
\fontconfig[alts]{%
  rm = texgyreschola,
  sf = texgyrebonum,
  tt = texgyreadventor,
  math = texgyrescholamath,
}
```

With this setup,

```
\usefontset{alts}
```

would switch to the second configuration and

```
\usefontset{default}
```

would switch back to the original setup. `\boldmath` may be used as usual while a font set is active; if a bold version is configured for the set, it will be used as usual[5]. Similarly, `\normalmath` switches to the 'normal' version of the current font set[6].

[3] If you use Unicode maths, note that lua-unicode-math *does* add fake bold if otherwise unavailable. This is mainly an issue for Latin Modern Math, since this is the only case where lua-font-config uses lua-unicode-math's font definitions.

[4] Where 'reasonable' means I have some idea what they do and some, possibly merely theoretical, ability to test them. This is naturally a quite unreasonable definition of 'reasonable' with a clear Western European bias. Input from users of non-Latin scripts (initially for mono-script documents) would be particularly welcome.

[5] In contrast, note that will switch to the default set's bold, which is probably not what you want.

[6] Again, will switch to the `normal` version of the default set, which is probably not what you want if a different font set is enabled.

3 Usage

There are currently three configuration commands: `\fontconfig` is the main command and the only one required in most documents. Documents which need to switch font configurations in the body of the document also require `\newfontset` and `\usefontset`. These commands are explained in section 3.3.

`\fontconfig` [⟨*font set*⟩]{⟨*key-val list*⟩}

Specify font configuration.

⟨*font set*⟩ is relevant only if multiple sets of fonts are required (see section 3.3). The default is to specify the document's default font configuration i.e. the one active at `\begin{document}`.

3.1 Keys

Top-level keys are described here. Keys for use in key-value lists are described in section 3.2.

`rm` (*key*) = ⟨*family name*⟩

`rm` (*key*) = {⟨*key-val list*⟩}

Specify family to use as the main document font, typically a serif or 'roman' family.

Available keys are listed below.

`sf` (*key*) = ⟨*family name*⟩

`sf` (*key*) = {⟨*key-val list*⟩}

Specify family to use as the default sans family, typically a sans-serif.

Available keys are listed below.

`tt` (*key*) = ⟨*family name*⟩

`tt` (*key*) = {⟨*key-val list*⟩}

Specify family to use as the default 'typewriter' family, typically monowidth.

Available keys are listed below.

`family` (*key*) = ⟨*family name*⟩

`family` (*key*) = {⟨*key-val list*⟩}

Specify an additional text family. This will create a macro which can be used to switch to the requested family[7].

Available keys are listed below.

`math` (*key*) = ⟨*family name*⟩

`math` (*key*) = ⟨*key-val list*⟩

Specify the family to use for typesetting Unicode mathematics. This sets the fonts used for symbols, script and calligraphic alphabets. It does not change the fonts used to typeset operators, since these are taken from text fonts and configured separately.

Available keys are listed below.

This option uses lua-unicode-math. unicode-math is **not** supported. If you wish to use unicode-math, do not use this option. Instead, load unicode-math and use `\setmathfont` as usual.

`debug` (*key*) = {⟨*key-val list*⟩}

[7] Usually you will want to specify which macro, since otherwise you must figure out the family name the code uses, which may or may not match the one requested. See section 3.2.4.

This can be used to enable debugging and/or modify the logging level. It is primarily intended for use in debugging the package and unlikely to be of use in production. However, it may be useful the use this key if reporting a bug which cannot be reproduced e.g. which is platform specific or depends on system fonts.

3.2 Keys for specifying font families

All keys described in section 3.1 accept either a simple family name, as shown in the example above, or a braced key-value list. For example,

```
\usepackage{lua-font-config}
\fontconfig{%
  rm = {%
    name = texgyrepagella,
    features  = {%
      script=ltn,
      figures={proportional,oldstyle},
      % tex ligatures are anyway default
      ligatures=tex,
    },
  },
  sf = {%
    name  = texgyreheros,
    scale = 1.2,
    features   = {%
      script=cyrl,
      figures={proportional,lining},
    },
  },
  tt = {%
    name  = texgyrecursor,
    features   = {%
      figures={tabular,lining,zero},
    },
    %% for completeness: false is anyway the default
    force = false,
  },
}
```

This section describes these 'inner' keys.

3.2.1 Common keys

These keys are accepted by all of `rm`, `sf`, `tt`, `family` and `math`.

name (*key*) = ⟨*lookup name*⟩

The 'name' to lookup. This may, but need not be, the name of a font family. Shorter, less specific names will configure more families; longer, more specific names will configure fewer.

For example, `name = texgyrepagellax`, will configure fewer than `name = texgyrepagella`, since the latter string matches both families, while the former does not.

The name is lower-cased prior to lookup. The data used to generate actual font families uses a cleaned up version of the given name or, sometimes, a variation, depending on the data returned from the database.

For example, `tex gyre PaGellA` yields the same results as `texgyrepagella`.

Numbers are likewise removed[8].

`force` (*key*) = `true` | `false`

Boolean which determines whether cached data is to be regenerated on the basis of a new search of the font-loader's database.

Note that this doesn't regenerate the loader's database itself. It simply zaps data stored by this module for one or more font families.

`scale` (*key*) = ⟨*factor*⟩

Optional factor by which the font should be scaled e.g. `20` or `0.8`. Ignored if a family includes distinct optical sizes.

3.2.2 Text-mode keys

These keys are accepted in all key-value specifications except for the top-level `math` key used to configure Unicode maths. They are used to configure font 'features' of various sorts. These include opentype features provided by the fonts, but also some additions provided by the font-loader or in other ways.

There are currently three keys in this category. One provides a general way to configure font features; two provide methods of last resort for faking bold and oblique fonts, when these are not available.

`features` (*key*) = {⟨*key-value list*⟩}

Optional list of font features **for text fonts only**.

Note that this interface is *highly* experimental: it may, and very likely will, change[9].

The default is equivalent to `features={ script=dflt, mode=node, language=dflt, ligatures=tex }`. Unless provided in separate fonts, small-caps are enabled, if available, for upright, italic and oblique small-caps.

Features apply to *all* fonts in the family specified by `name`, so you should not specify features you do not wish applied to all of them.

Currently, nine keys are supported. Where multiple values are listed, any combination of values is permitted in any order, but **inconsistent values should be avoided**.

A ⟨*key list*⟩ requires a comma-separated list of keys *without values*:

{⟨*key1*⟩,⟨*key2*⟩,...}

A ⟨*key-value list*⟩ requires a comma-separated list of key-values:

{⟨*key1*⟩=⟨*value1*⟩,⟨*key2*⟩=⟨*value2*⟩,...}

If both ⟨*key*⟩ and no⟨*key*⟩ are listed, no⟨*key*⟩ disables the feature and ⟨*key*⟩ enables it.

For example, the following disables TEX, 'required' and 'common' ligatures for the typewriter family and explicitly enables them for the roman/main/serif family.

```
\fontconfig{%
  tt = {%
    name = TeX Gyre Cursor,
    ligatures = {notex,norequired,nocommon},
```

[8] This is problematic for fonts which depend on digits in their names to distinguish them e.g. Venturis ADF has four distinct 'titling' fonts, whose names differ by a single character, which is 1, 2, 3 or 4. The algorithm used to sort the fonts into NFSS families currently gets confused by this, leading to misleading warnings about 'apparent duplicate' fonts. (Something is indeed awry, but they are not duplicates. Rather what should be four families appears to be only one.)

[9] It is almost certain that it excludes things people need; it may also include things they don't. Moreover, the groups of options are somewhat arbitrary, as are the keys' names. It needs significant user input to determine what works best here, as fontspec's framework is quite different.

```
  },
  rm = {%
    name = TeX Gyre Schola,
    ligatures = {tex,required,common},
  },
}
```

Generally, it is not necessary to use the positive form of keys in these cases, as the no⟨*key*⟩ is provided precisely so that users can override features enabled by default (typically by the font loader or LaTeX).

`alternates` (*key*) = {⟨*key/key-value list*⟩}

⟨*key/key-value list*⟩ should be a comma-separated list of one or more of the following valueless keys

Valueless key	Controls
`contextual`|`nocontextual`	standard contextual alternates
`contextualswash`	contextual 'swash'
`linefinal`	end-of-line forms
`wordfinal`	end-of-word forms
`justification`	forms designed to help justify text
`swash`	'swash'

and/or one or more of the following key-value pairs

key	= Value	Activates
`annotation`	= ⟨*integer*⟩	'annotation' form ⟨*integer*⟩
`charactervariant`	= ⟨*integer i*⟩:⟨*integer j*⟩	the *i*th character mapping, selecting the *j*th variant
`stylistic`	= ⟨*integer*⟩	'stylistic' form ⟨*integer*⟩
`sset`	= ⟨*integer*⟩	the set of 'stylistic' forms ⟨*integer*⟩

`embolden` (*key*) = ⟨*factor*⟩

Add 'fake bold' to make the family heavier. Note that this applies to all fonts in the family; it is not intended to add fake bold to a family which lacks one.

`figures` (*key*) = {⟨*key list*⟩}

⟨*key list*⟩ should be a comma-separated list of one or more of the following valueless keys

Valueless key	Activates
`lining`	lining digits
`osf`|`oldstyle`	oldstyle / hanging digits
`prop`|`proportional`	proportional digits
`tab`|`tabular`	tabular digits
`zero`	the slashed zero

`forms` (*key*) = {⟨*key list*⟩}

⟨*key list*⟩ should be a comma-separated list of one or more of the following valueless keys

Valueless key	Effect
capstopetite	map ABC to ‘petite’ caps
capstosc	map ABC to small-caps
case	enable forms designed for use in capitalised contexts
nocase	disable them
hist|historical	enable ‘historical’ forms of characters
random	characters vary pseudo-randomly
smalltopetite	map abc to ‘petite’ caps
titling	enable forms intended for use in titles
unicase	map ABCabc to letters in a single-cased alphabet

hyphenchar (*key*) = true|false

Enables or disables hyphenation for the family. Generally, this key should not be required except, possibly, to disable hyphenation for the ‘typewriter’ family, if this is used for code listings.

For example, this document is typeset with the following configuration:

```
\usepackage{lua-font-config}
\fontconfig{%
  rm    = kproman,
  sf    = kpsans,
  tt    = {%
    name  = kpmono,
    % mode, script and language are unnecessary as these are anyway default
    % -tlig disables TeX ligatures
    fea   = {mode=node;script=dflt;language=dflt;-tlig},
    % disable hyphenation as we don't want that in code listings
    hyphenchar  = false,
  },
  math  = kpmath,
}
```

language (*key*) = ⟨*language id*⟩

⟨*language id*⟩ must currently be a valid identifier, as recognised by luaotfload. `otfinfo -l` ⟨*path to font file*⟩ may be used to determine identifiers supported by the font.

ligatures (*key*) = {⟨*key list*⟩}

⟨*key list*⟩ should be a comma-separated list of one or more of the following valueless keys

Valueless key	Controls
contextual|nocontextual	‘contextual’ ligatures
discretionary	discretionary ligatures
hist|historic	historical ligatures
required|norequired	required ligatures
common|nocommon	‘common’ ligatures
tex|notex	TEX ligatures e.g. --

mode (*key*) = base|node|harf|harfbuzz

Set the rendering mode.

script (*key*) = ⟨*script id*⟩

⟨*script id*⟩ must currently be a valid identifier, as recognised by luaotfload. `otfinfo -s` ⟨*path to font file*⟩ may be used to determine identifiers supported by the font.

scope (*key*) = {⟨*key-value list*⟩}

Limit effect of current list of features to fonts satisfying the conditions specified by the ⟨*key-value*⟩ list.

This key is special: it does not itself provide access to font features; instead, it limits the application of features specified using the other keys.

This key should be used with caution. It should be used only where an option only makes sense for certain fonts e.g. special spacing for small-caps. It is not designed to support hodgepodge families of random sets of features applied to different members.

`weight` (*key*) = ⟨*weight selector*⟩

`width` (*key*) = ⟨*width selector*⟩

`case` (*key*) = ⟨*case selector*⟩

`style` (*key*) = ⟨*style selector*⟩

⟨*key-value list*⟩ should be a comma-separated list of one or more of these keys with the desired selector(s). Supported selectors:

weight	width	case	style	
`ultrabold`	`ultraexpanded`			
`extrabold`	`extraexpanded`			
`semibold`	`semiexpanded`			
`demibold`				
`bold`	`expanded`			
`medium`				
`book`				
`regular`	`medium`	`mixed`	`upright`	default
`semilight`	`semicondensed`	`smallcaps`	`italic`	
`light`	`condensed`		`oblique`	
`extralight`	`extracondensed`			
`ultralight`	`ultracondensed`			

Note that `medium` is *not* a synonym for `regular`. If you aren't sure which you need, try `regular`.

`spacing` (*key*) = {⟨*key list*⟩}

⟨*key list*⟩ should be a comma-separated list of one or more of the following valueless keys

Valueless key	Controls
`capsspacing`	spacing designed for capitalised text
`kern`|`nokern`	kerning

`fakebold` (*key*) = ⟨*factor*⟩

`fakeoblique` (*key*) = ⟨*factor*⟩

Add fake bold and/or oblique to families which lack them.

⟨*factor*⟩ is optional in both cases. The default values are `1.5` and `0.2` respectively. These values were shamelessly stolen from fontspec.

Note that this is always a bad idea, in my opinion, and I would personally never use this in production. Even those more sympathetic to such methods typically agree it is *almost* always a bad idea. Fonts reflect designers' aesthetic and artistic conceptions, the typographic norms of the cultures and periods in which they are created and their intended applications. Adding bold to a garamond or oblique to IM Fell English is typographically monstrous and the results should never be used in production.

Consider these features ones provided for outrageous typographic expressions, typesetting shock tactics, font-based rebellions and the like.

3.2.3 Default text family keys

These keys are accepted by `rm`, `sf` and `tt` i.e. when specifying the default text families.

These first four keys all do the same thing, so you may use your preferred syntactic sweetener.

`math` (*key*) = `true`|`false`
`maths` (*key*)

Should the current specification also be used for the corresponding text family in maths?

For example, if you are configuring the main document family with `rm`, should that family also be used for operators, `\mathrm` etc.?

The default is equivalent to `math = true`[10].

`nomath` (*key*) Equivalent to `math = false`.
`nomaths` (*key*)

`bold` (*key*) = {⟨*key-value list*⟩}

`normal` (*key*) = {⟨*key-value list*⟩}

Respectively set the default used for bold (`\bfseries`) and regular/normal (`\mdseries`).

Each ⟨*key-value list*⟩ may take one or both of the following keys:

`weight` (*key*) = ⟨*weight selector*⟩

`width` (*key*) = ⟨*width selector*⟩

Supported selectors:

Weight selectors	Width selectors
`ultrabold`	`ultraexpanded`
`extrabold`	`extraexpanded`
`bold`	`expanded`
`semibold`	`semiexpanded`
`demibold`	
`medium`	
`book`	
`regular`	`medium`
`semilight`	`semicondensed`
`light`	`condensed`
`extralight`	`extracondensed`
`ultralight`	`ultracondensed`

Advanced users may alternatively set the NFSS series directly:

`bfseries` (*key*) = ⟨*series specifier*⟩

`mdseries` (*key*) = ⟨*series specifier*⟩

Series are those *and only those* documented as supported by LaTeX's NFSS[11].

[10] If configuring a non-default font set, the default is `false` unless the set explicitly specifies a Unicode maths font with the `math` key, in which case, it defaults to `true`. *It is strongly recommended to give the desired value of this option explicitly for non-default font sets, regardless of whether that value happens to coincide with the current default, which may change without notice.*

[11] If you wish to specify 'book' or 'medium' for a font which also has a regular (and/or book) weight, please use the `normal` and/or `bold` keys. These weights do not have a standard representation distinct from `m`. This is a limitation of the schema provided by LaTeX's font management. Ideally, LaTeX would provide symbolic representations of these weights, since many fonts now have them *as well as* regular, rather than instead. However, it doesn't, so lua-font-config does nefarious things which innocent readers, even the expert ones, should not wish to know. More seriously, the implementation will change without notice if I find a way to avoid the nefarious. If you really want to ignore this warning, see `scoped` in section 3.4. *Only series documented in LaTeX's `fntguide` are supported.*

3.2.4 Additional text family keys

These keys are accepted by `family` i.e. when specifying additional non-default text families.

command (*key*) =\command

Use `\command` to switch to the font family, rather than creating a command from the family name. **Highly recommended.**

By default the command to switch to the font family depends on the discovered family name i.e. on the result of searching the database for the specified family. In many cases, this will be the value passed to the `name` key, but often it will not and sometimes this may not be the command name you want anyway. `command` allows you to override it.

For example, the following would setup `\playfair` to switch to Playfair Display, rather than the default choice of `\playfairdisplay`.

```
\usepackage{lua-font-config}
\fontconfig{%
  family = {% setup an additional non-default family
    name = Playfair Display,
    command = \playfair,
  },
}
```

3.3 Font sets

Font sets allow multiple, independent sets of fonts to be used in a document, in such a way that standard commands such as `\textsf`, `\bfseries` and `\boldmath` 'just work'[12].

Name: font set

Concept: A complete set of fonts which may be activated at any point in the body of the document.

Behaviour: At any point during typesetting, exactly one font set should be active; activating an alternative font set should deactivate the current one. While active, a font set determines the effect of all standard font selection commands for both `text` and `math` modes.

Default: The `default` font set should always be present and active at the start of the document body.

Configuration: Font sets should be configured independently of each other. The effect of switching to a different font set should produce the same behaviour as configuring that font set for the whole document.

Components: A font set consists of the following:

- A unique name. `default` is reserved for the default font set. There are some restrictions on names, but any other combination of lowercase Latin letters is certainly safe. Names must be unique.
- A serif/roman/main family, either specified using the `rm` key (sections 3.1 and 3.2) or `lmr`.
- A sans family, either specified using the `sf` key (sections 3.1 and 3.2) or `lmss`.
- A typewriter/monowidth/mono family, either specified using the `tt` key (sections 3.1 and 3.2) or `lmtt`.
- A Unicode maths family, together with families for script and calligraphic alphabets, either specified using the `math` key (sections 3.1 and 3.2) or `lmm`.

[12] Thanks to David Carlisle for guidance implementing this feature.

- Per-family weights for the default and bold series used in text (`\textbf` etc.).
- `normal` and `bold` versions for mathematics, *provided these exist and are accessed using `\normalmath` and `\boldmath`.*

Purpose: Documents which require multiple font configurations e.g. multilingual documents involving different scripts, features etc., older texts with contemporary commentaries, where the primary text should reflect typesetting conventions which have since changed (long 's', 'ß') or documents which use the same mathematical characters with differences in meaning and spacing. Font sets are intended to allow easy setup of, and seamless switches between, the various configurations necessary.

Font sets[13] require lua-unicode-math. The package will be automatically loaded, if necessary, or an error produced, if the incompatible unicode-math is already loaded.

3.3.1 A schematic example

To configure font sets `seta` and `setb`, proceed as follows:

```
\usepackage{lua-font-config}
% The default font set **MUST** be configured first.
\fontconfig{%
  % Specify **at least one** of rm, sf, tt and math.
  % It is fine to specify a LaTeX default if you don't want to change it.
  rm = lmr,
  % Specify any additional special families required.
  % family = {...},
}
% Initialise our additional font sets.
\newfontset{seta}
\newfontset{setb}
% Configure font set 'seta'.
\fontconfig[seta]{%
  % Specify **at least one** of rm, sf, tt, math.
  ...
}
% Configure font set 'setb'.
\fontconfig[setb]{%
  % Specify **at least one** of rm, sf, tt, math.
  ...
}
```

The `default` set will be active when the document begins. Use

```
\usefontset{seta}
```

to switch to `seta`,

```
\usefontset{setb}
```

to activate `setb` and

```
\usefontset{default}
```

to restore the `default` set.

[13] Technically, lua-font-config always uses font sets, but most documents require only the `default` font set. In this case, lua-unicode-math is not required (unless the top-level `math` key is used).

3.3.2 Commands

Use `\newfontset` to initialise font sets, `\fontconfig` to configure them and `\usefontset` to activate them in your document.

`\newfontset` [⟨*base-identifier*⟩]{⟨*new-identifier*⟩}

Initialise a font set ⟨*new-identifier*⟩, optionally based on an existing font set ⟨*base-identifier*⟩.

Apart from the `default` font set, which is initialised automatically, all font sets must be initialised using this command prior to configuration.

`\fontconfig`[⟨*identifier*⟩]{⟨*key-value list*⟩} should then be used to configure the font set in the usual way (section 3).

All font sets must be configured with `\fontconfig` before they may be used.

`\newfontset` and `\fontconfig` are available in the preamble of the document.

`\usefontset` {⟨*identifier*⟩}

Switch to font set ⟨*identifier*⟩.

This command is available at any point in the body of the document.

`\usefontset`{⟨*default*⟩} may be used to restore the default font set.

`\normalmath` Switch to the `normal` maths version of the current font set.

`\boldmath` Switch to the `bold` maths version of the current font set.

`\boldmath` is redefined by `\usefontset` to adjust the result to the active configuration.

`\normalmath` is similarly redefined, but since it is not provided by LaTeX, a suitable definition is provided by the package as a baseline. Although this command is documented here, where it seems potentially of some interest, it is provided whether additional font sets are used or not[14].

3.4 Advanced configuration

`fea` and `scoped` provide a relatively lower-level interface to the functionality provided by the `features` key. They apply to the configuration of text families only and require familiarity with both NFSS, luaotfload and opentype feature tags.

These keys are faster and more powerful than the keys described earlier. They may be used in the same places as `features` (section 3.2.2).

If you are not sure which to use or lack familiarity with NFSS, luaotfload or opentype feature tags, use `features`.

`fea` and `features` may be used to configure the same family. If so, consistent options are combined; in case of conflicts, `features` takes precedence over `fea`. This means you can use the more-user friendly interface for most things and still add any unsupported specialist setup, if necessary.

`fea` (*key*) = {⟨*opentype feature spec.*⟩}

Optional list of font features **for text fonts only**.

Note that the specification is passed to the font loader with minimal processing, so you must use 'raw' feature specifications. *If you do not know what these are, use the `features` key rather than this one; if you need an unsupported feature, file an issue on the bug tracker (link below).*

The default is equivalent to `fea={mode=node;scrpt=dflt;language=dflt;+tlig}`. Unless provided in separate fonts, `+smcp` is added, if available, for upright, italic and oblique small-caps.

[14] In fact, it is provided even if nothing is configured with `\fontconfig`, so that the document fonts are untouched. In that case, it just works analogously to `\boldmath`.

Features apply to *all* fonts configured as a result of the lookup, so you should not specify features you do not wish applied to all fonts to which the specification applies.

For example,

```
\usepackage{lua-font-config}
\fontconfig{%
  rm = {%
    name = texgyrepagella,
    fea  = {script=ltn;+tlig;+pnum},
    % apply 'capital spacing' to small-caps upright/italic/oblique
    scoped = {*:sc}{+cpsp},
    scoped = {*:scit}{+cpsp},
    scoped = {*:scsl}{+cpsp},
  },
  sf = {%
    name  = texgyreheros,
    scale = 1.2,
    fea   = {script=cyrl;},
  },
  tt = {%
    name  = texgyrecursor,
    fea   = {+zero;},
    %% for completeness: false is anyway the default
    force = false,
  },
}
```

scoped (*key*) = {⟨*scope spec.*⟩}{⟨*opentype feature spec.*⟩}

Optional list of font features **for a subset of text fonts only**.

⟨*scope spec.*⟩ is a colon-separated expression consisting of a series specifier followed by a shape specifier i.e. ⟨*series*⟩:⟨*shape*⟩. One expression may be * meaning all series or all shapes.

For example, `scoped=*:sc+cpsp` would add 'capital spacing' to all small-caps fonts, regardless of series.

⟨*series*⟩ and ⟨*shape*⟩ have take the usual `nfss` values, except that `book` is `k`, if a family also has a regular variant, and `medium` is `f`, if a family also has either book or regular. **These specifiers are non-standard and may change without notice, if I can think of a way to avoid them.**

Features apply *only* to fonts covered by ⟨*scope spec.*⟩ The ⟨*opentype feature spec.*⟩ is *added* to the features applied to all fonts configured by the lookup. So, if you want oldstyle figures for all fonts except bold oblique, you must add `+lnum` for the `b:sl` scope.

Note that both the scope and feature specifications are passed to LaTeX and the font loader with minimal processing, so you must use low-level series/shape representations and 'raw' feature specifications. *If you do not know what these are, use the `features` key rather than this one; if you need an unsupported feature, file an issue on the bug tracker (link below).*

Note that although arbitrary scopes and features may minimally work, **this is not recommended**. The intended usage is to change spacing or figures in small-caps variants (see fig. 9), say, or to substitute petite for small caps or make similar adjustments. In particular, note that the underlying family names become unpredictable and possibly inconsistent if too many changes create a hodgepodge rather than a coherent family. Moreover, documents typeset with such fonts will generally be typographically inconsistent, too. If more variety is really needed, create additional font families and switch between them rather than abusing `scoped`.

Note that this key cannot be used to add 'fake' fonts: you cannot add a fake bold or oblique this way. Use `fakebold`, `fakeoblique` etc. (section 3.2.2) for this, if necessary.

4 Compatibility

4.1 fontspec

The package should be compatible with fontspec, provided particular fonts are configured with commands from one or neither package. However, this is not tested and obviously documents cannot benefit from any performance gains to the extent they rely on fontspec[15].

4.2 ltx-talk

Load with `\RequirePackage` *before* loading the class. The package emulates the class settings by default, so the main benefit of loading it is speed. If you wish to change the fonts, use `\fontconfig` after the class is loaded[16].

4.3 lua-unicode-math

lua-font-config is definitely compatible with lua-unicode-math and will load it if the top-level `math` key is used for Unicode maths or a non-default font set is created. Note that if you do use the `math` key and in any case for additional font sets, none of the `.fd` or font-specific `.sty` files will be used; instead, the maths fonts will be configured using the same process as the text fonts and results cached in the same way. If you would prefer to use the `.fd` files from lua-unicode-math, but don't want to load one of the font-specific packages it provides, you can do so by using the correct family name in `\fontconfig`. For example, the following should use lua-unicode-math's font definitions from `tueuler-math.fd`, `tueuler-math-sf.fd` and `tueuler-math-ssf.fd` for Euler Math, without loading lum-euler.

```
\fontconfig{%
  math = euler-math,
}
```

In general, lua-font-config will read existing font definitions, where these are available, but this requires an exact name match.

4.4 unicode-math

The package should be compatible with unicode-math, provided neither the top-level `math` key (for Unicode maths configuration) nor (non-default) font sets are used. See section 4.1, but there is really no performance reason to use lua-font-config in this case, since unicode-math is extremely slow and this will easily dominate the time taken for font configuration. In general, I see no reason to use lua-font-config together with unicode-math; if you need the latter, there is really no reason to use lua-font-config at all.

5 Examples

Figures 1 to 3 demonstrate simple use of the package to configure the main document font, a Unicode maths font (*via* lua-unicode-math and a complete font configuration (roman/serif, sans, typewriter and maths). These examples use the default configurations for text and maths fonts, so the configurations are extremely simple.

Figure 4 demonstrates how to use lua-font-config to speed up the compilation of presentations created with ltx-talk.

Figures 5 to 7 demonstrate slightly more complex setups.

Figure 8 demonstrates how to configure alternative sets of fonts and how to switch sets in the document.

[15] But fontspec's performance is quite good if you do not use fonts with `.fontspec` files or which are especially large. E.g. Latin Modern and TEX-Gyre fonts are generally fine, but New Computer Modern is rather painful.

[16] If this turns out to be something people want to do, I can adjust things to avoid configuring fonts twice. For now, this is just a good way for me to test the code and avoid the pain of watching fontspec load New Computer Modern.

Figure 9 demonstrates use of a lower-level interface to enable features for a customisable subset of fonts in a family.

example-lmr -- fig. 1

```
\documentclass{article}
\usepackage{lua-font-config}
\fontconfig{rm = latinmodernroman}
\begin{document}
This is Latin Modern Roman.
It features \textit{italics} and \textsl{oblique}.
\textsc{Small-caps are implemented as distinct fonts, as are \textsl{oblique
↪ small-caps}.}
{\fontshape{ui}\selectfont Upright italics are less commonly used.}

\normalfont\bfseries
This is the bold expanded weight.
It features \textit{italics} and \textsl{oblique}.

\normalfont\fontseries{db}\selectfont
The ‘demi-bold’ is of regular width, with \textsl{oblique}.

\normalfont
\Huge
Where optical sizes are available, these provide higher quality output than simple
↪ scaling.

\LARGE
Where optical sizes are available, these provide higher quality output than simple
↪ scaling.

\Large
Where optical sizes are available, these provide higher quality output than simple
↪ scaling.

\large
Where optical sizes are available, these provide higher quality output than simple
↪ scaling.

\normalsize
Where optical sizes are available, these provide higher quality output than simple
↪ scaling.

\small
Where optical sizes are available, these provide higher quality output than simple
↪ scaling.

\footnotesize
Where optical sizes are available, these provide higher quality output than simple
↪ scaling.

\scriptsize
Where optical sizes are available, these provide higher quality output than simple
↪ scaling.

\tiny
Where optical sizes are available, these provide higher quality output than simple
↪ scaling.
```

This is Latin Modern Roman. It features *italics* and *oblique*. Small-caps are implemented as distinct fonts, as are *oblique small-caps*. Upright italics are less commonly used.
This is the bold expanded weight. It features *italics* and *oblique*.
The 'demi-bold' is of regular width, with *oblique*.

Where optical sizes are available, these provide higher quality output than simple scaling.

Where optical sizes are available, these provide higher quality output than simple scaling.

Where optical sizes are available, these provide higher quality output than simple scaling.

Where optical sizes are available, these provide higher quality output than simple scaling.

Where optical sizes are available, these provide higher quality output than simple scaling.

Where optical sizes are available, these provide higher quality output than simple scaling.

Where optical sizes are available, these provide higher quality output than simple scaling.

Where optical sizes are available, these provide higher quality output than simple scaling.

This is footnote-sized Latin Modern Roman. It features *italics* and *oblique*. Small-caps are implemented as distinct fonts, as are *oblique small-caps*. Upright italics is less commonly used.
This is the bold expanded weight. It features *italics* and *oblique*.
The 'demi-bold' is of regular width, with upright and *oblique* shapes.
The output should look very much as if 'lua-font-config' had not been loaded!

Figure 1: `example-lmr`

```
\normalsize
\footnotesize
This is footnote-sized Latin Modern Roman.
It features \textit{italics} and \textsl{oblique}.
\textsc{Small-caps are implemented as distinct fonts, as are \textsl{oblique
↪  small-caps}.}
{\fontshape{ui}\selectfont Upright italics is less commonly used.}

\normalfont\bfseries
This is the bold expanded weight.
It features \textit{italics} and \textsl{oblique}.

\normalfont\fontseries{db}\selectfont
The 'demi-bold' is of regular width, with upright and \textsl{oblique} shapes.

\normalfont
The output should look very much as if 'lua-font-config' had not been loaded!
\end{document}
```

example-ncm-math -- fig. 2

```
\documentclass{article}
```


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

$$\sin^2\theta + \cos^2\theta = 1$$

xxx

$$\mathbf{\sin^2\theta + \cos^2\theta = 1}$$

xxx

Figure 2: example-ncm-math

```
\usepackage{lua-font-config}
\fontconfig{%
  rm = newcomputermodern,
  sf = newcomputermodernsans,
  tt = newcomputermodernmono,
  math = newcomputermodernmath,
}
\parindent=0pt
\begin{document}
\[
    \sin^2\theta + \cos^2\theta = 1
\]
$\mathsf{x} \mathtt{x} \mathbf{x}$

\mathversion{bold}
\[
    \sin^2\theta + \cos^2\theta = 1
\]
$\mathsf{x} \mathtt{x} \mathbf{x}$
\end{document}
```

example-ncm-complete -- fig. 3

```
\documentclass{article}
\usepackage{lua-font-config}
\fontconfig{%
  rm    = New Computer Modern,
  sf    = New Computer Modern Sans,
  tt    = New Computer Modern Mono,
  math  = New Computer Modern Math,
}
\begin{document}
New Computer \textsl{Modern} \textit{with} \textsc{\textit{default} settings}.

\sffamily
New Computer \textsl{Modern} \textit{with} \textsc{\textit{default} settings}.

\ttfamily
New Computer \textsl{Modern} \textit{with} \textsc{\textit{default} settings}.

\normalfont
New Computer Modern Math with text fonts as above:
\[
    \sin^2\theta + \cos^2\theta = 1
\]
\[
    \mathscr{ABC abc} \mathcal{ABC abc}
```

New Computer *Modern with* DEFAULT SETTINGS.
New Computer *Modern with* DEFAULT SETTINGS.
New Computer *Modern with* DEFAULT SETTINGS.
New Computer Modern Math with text fonts as above:

$$\sin^2\theta + \cos^2\theta = 1$$

$$\mathscr{ABCabcABCabc}$$

$$\mathbf{\sin^2\theta + \cos^2\theta = 1}$$

$$\boldsymbol{\mathscr{ABCabcABCabc}}$$

Figure 3: example-ncm-complete

```
\]
\mathversion{bold}
\[
    \sin^2\theta + \cos^2\theta = 1
\]
\[
    \mathscr{ABC abc} \mathcal{ABC abc}
\]
\end{document}
```

Figure 4 demonstrates the default configuration for ltx-talk. This is approximately equivalent to the default setup if lua-font-config is not loaded. Note the way the package is loaded *before* \documentclass. If we used \usepackage after the class was loaded, we would not benefit from the faster compilation our Lua-based approach provides[17]. See section 4.2.

example-talk -- fig. 4

```
\DocumentMetadata{tagging=on,pdfstandard=ua-2,pdfversion=2.0,lang=en-GB}
\RequirePackage{lua-font-config}% MUST precede class.
\documentclass{ltx-talk}

\title{Some Title}
\author{A. U. Thor}
\institute{Pantheon, Greece}
\date{Ancient}
\begin{document}
\begin{frame}
  \maketitle
\end{frame}
\begin{frame}
  \frametitle{Frame}
  Some text\footnote{With a footnote.} on a slide.

  \textsc{Small-Caps \textsl{italics \textbf{bold}} \textbf{bold}.}

  Regular, \fontseries{k}\selectfont Book and \bfseries Bold.
  \normalfont
  \begin{equation}
    v = ut + \frac{1}{2}at^2
  \end{equation}
```

[17] In fact, although lua-font-config *is* faster generally and although this is likely due to greater use of Lua, the size of the difference in this particular case suggests some additional factor is involved. However, I've not tried to confirm this or to identify the cause.

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

```
  \begin{equation}
    \sin^2\theta + \cos^2\theta = 1
  \end{equation}
  Script $\mathscr{ABC abc}$ and calligraphic $\mathcal{ABC abc}$.

  Bold maths:
  \mathversion{bold}
  \begin{equation}
    v = ut + \frac{1}{2}at^2
  \end{equation}
  \begin{equation}
    \sin^2\theta + \cos^2\theta = 1
  \end{equation}
  Script $\mathscr{ABC abc}$ and calligraphic $\mathcal{ABC abc}$.
\end{frame}
\end{document}
```

Note that figs. 5 and 6 give exactly the same output; the only difference is that one uses `features` and the other `fea` to configure features for the configured families. See section 3.2.2.

example-features -- fig. 5

```
\documentclass{article}
\parindent=0pt
\usepackage{lua-font-config}
\fontconfig{%
  rm = {% serif uses oldstyle, proportional figures
    name = TeX Gyre Pagella,
    features = {%
      figures = {oldstyle, proportional},
    },
  },
  sf = {% sans uses proportional lining figures
    name = texgyreheros,
    features = {%
      figures = {lining, proportional},
    },
  },
  tt = {% typewriter uses tabular lining figures and no ligatures
    name = texgyrecursor,
    features = {%
      figures = {lining, tabular},
      ligatures = {notex, nocommon},
    },
  },
}
\begin{document}
{%
  \TeX-Gyre Pagella: 0123456789
}

{%
  \sffamily
  \TeX-Gyre Heros: 0123456789
}

{%
  \ttfamily
  \TeX-Gyre Cursor 0123456789
```

Some Title

A. U. Thor
Pantheon, Greece
Ancient

Frame

Some text[1] on a slide.
SMALL-CAPS *ITALICS* ***BOLD*** **BOLD**.
Regular, Book and **Bold.**

$$v = ut + \frac{1}{2}at^2 \tag{1}$$

$$\sin^2\theta + \cos^2\theta = 1 \tag{2}$$

Script $\mathscr{ABC}abc$ and calligraphic $\mathcal{ABC}abc$.
Bold maths:

$$v = ut + \frac{1}{2}at^2 \tag{3}$$

$$\boldsymbol{\sin}^2\theta + \boldsymbol{\cos}^2\theta = 1 \tag{4}$$

Script $\mathscr{ABC}abc$ and calligraphic $\mathcal{ABC}abc$.

[1] With a footnote.

Figure 4: `example-talk`

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

TEX-Gyre Pagella: 0123456789
TEX-Gyre Heros: 0123456789
TEX-Gyre Cursor 0123456789 -- --- outfitted officials affluently affiliate fish finials flagrantly, first finalising infinite floral affinities of fish-filled fingers

$\sin\theta$ rm sf tt *it* **bf**

$\sin\theta$ rm sf tt *it* bf

Figure 5: example-features

```
  -- --- outfitted officials affluently affiliate fish finials flagrantly, first
  finalising infinite floral affinities of fish-filled fingers
}

\[
    \sin\theta\quad
    \mathrm{rm}\quad
    \mathsf{sf}\quad
    \mathtt{tt}\quad
    \mathit{it}\quad
    \mathbf{bf}
\]
\mathversion{bold}
\[
    \sin\theta\quad
    \mathrm{rm}\quad
    \mathsf{sf}\quad
    \mathtt{tt}\quad
    \mathit{it}\quad
    \mathbf{bf}
\]
\end{document}
```

example-fea -- fig. 6

```
\documentclass{article}
\parindent=0pt
\usepackage{lua-font-config}
\fontconfig{%
  rm = {% serif uses oldstyle, proportional figures
    name = TeX Gyre Pagella,
    fea = {+ONUM;+PNUM},
  },
  sf = {% sans uses proportional lining figures
    name = texgyreheros,
    fea = {+pnum;+lnum},
  },
  tt = {% typewriter uses tabular lining figures and no ligatures
    name = texgyrecursor,
    fea = {-TLIG;+tnum;+lnum;-liga},
  },
}
\begin{document}
{%
  \TeX-Gyre Pagella: 0123456789
```

TEX-Gyre Pagella: 0123456789
TEX-Gyre Heros: 0123456789
TEX-Gyre Cursor 0123456789 -- --- outfitted officials affluently affiliate fish finials flagrantly, first finalising infinite floral affinities of fish-filled fingers

$\sin\theta$ rm sf tt *it* **bf**

$\sin\theta$ rm sf tt *it* bf

Figure 6: example-fea

```
}

{%
  \sffamily
  \TeX-Gyre Heros: 0123456789
}

{%
  \ttfamily
  \TeX-Gyre Cursor 0123456789
  -- --- outfitted officials affluently affiliate fish finials flagrantly, first
  finalising infinite floral affinities of fish-filled fingers
}

\[
    \sin\theta\quad
    \mathrm{rm}\quad
    \mathsf{sf}\quad
    \mathtt{tt}\quad
    \mathit{it}\quad
    \mathbf{bf}
\]
\mathversion{bold}
\[
    \sin\theta\quad
    \mathrm{rm}\quad
    \mathsf{sf}\quad
    \mathtt{tt}\quad
    \mathit{it}\quad
    \mathbf{bf}
\]
\end{document}
```

Figure 7 demonstrates usage of the fam key to configure an additional family and a custom font switch to access it.

example-add -- fig. 7

```
\documentclass{article}
\usepackage{lua-font-config}
\fontconfig{%
  family  = {
    name    = Lobster Two,
    scale   = 1.2,
    command = \lobster,
```

This is some ordinary text in the default LaTeX 2_ε font.

This is Lobster Two.
Lobster Two italics.
Lobster Two bold.
Lobster Two bold italics.

Back to the default.

Figure 7: `example-add`

```
  },
}
\begin{document}
This is some ordinary text in the default \LaTeXe{} font.\bigskip

{\lobster This is Lobster Two.\par
  \textit{Lobster Two italics.}\par
  \textbf{Lobster Two bold.}\par
  \textit{\textbf{Lobster Two bold italics.}}\par
}
\bigskip

Back to the default.
\end{document}
```

Figure 8 demonstrates usage of font sets to switch between multiple font configurations (section 3.3).

example-fontsets -- fig. 8

```
\documentclass{article}
\usepackage{lua-font-config}
\fontconfig{%
  rm = {%
    name = TeX Gyre Pagella,
    features = {figures = {oldstyle,proportional},},
  },
  sf = {% We could omit lining here as it is the default for these fonts.
    name = TeX Gyre Heros,
    features = {figures = {lining,proportional},},
  },
  tt = {% Avoid ligatures in code listings etc.
    name = TeX Gyre Cursor,
    features = {figures = {lining,tabular}, ligatures = {nocommon,notex},},
  },
  math = TeX Gyre Pagella Math,
}
\newfontset{latinii}% Create a new font set named 'latinii'.
\fontconfig[latinii]{% Configure our alternative font set.
  rm = {% We use lining, rather than oldstyle, figures for this set.
    name = TeX Gyre Schola,
    features = {figures = proportional},
  },
  sf = {%
    name = TeX Gyre Adventor,
    features = {figures = proportional},
  },
  tt = {% We repeat our typewriter setup - w/o this, we'd get lmtt.
```

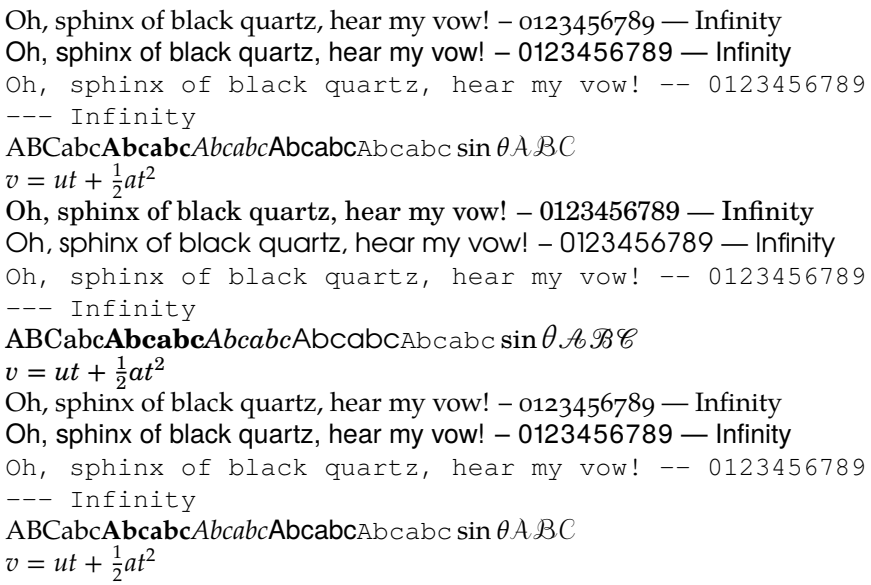


Figure 8: example-fontsets

```
    name = TeX Gyre Cursor,
    features = {figures = {lining,tabular}, ligatures = {nocommon,notex},},
  },
  math = TeX Gyre Schola Math,
}
\newcommand*\testme{Oh, sphinx of black quartz, hear my vow! -- 0123456789 ---
↪  Infinity\par}
\newcommand*\testma{%
  $\mathrm{ABC abc} \mathbf{Abc abc} \mathit{Abc abc} \mathsf{Abc abc} \mathtt{Abc
  ↪  abc} \sin\theta \mathscr{ABC}$\par
  $v = ut + \frac{1}{2}at^2$\par
}
\begin{document}
\rmfamily \testme
\sffamily \testme
\ttfamily \testme
\normalfont \testma

\usefontset{latinii}
\rmfamily \testme
\sffamily \testme
\ttfamily \testme
\normalfont \testma

\usefontset{default}
\rmfamily \testme
\sffamily \testme
\ttfamily \testme
\normalfont \testma
\end{document}
```

Figure 9 demonstrates usage of the low-level scoped key to enable features selectively.

example-scoped -- fig. 9

```
\documentclass{article}
\usepackage{lua-font-config}
```

Oh, sphinx of black quartz, judge my vow!
Oh, sphinx of black quartz, judge my vow!
Oh, sphinx of black quartz, judge my vow!
Oh, sphinx of black quartz, judge my vow!

Figure 9: `example-scoped`

```
\fontconfig{%
  family = {%
    name = texgyrepagella,
    command = \tgpag,
    fea = {script=latn;+pnum;+onum},
    scoped = {*:sc}{+cpsp},
  },
}
\begin{document}
\tgpag Oh, sphinx of black quartz, judge my vow!

\textbf{Oh, sphinx of black quartz, judge my vow!}

\scshape
Oh, sphinx of black quartz, judge my vow!

\textbf{Oh, sphinx of black quartz, judge my vow!}
\end{document}
```

6 Implementation

There was very little here, because almost everything is done in Lua. While there is more here now, it is still mostly a wrapper around the Lua.

expl3 function names use the standard syntax conventions. In the source a function which begins __lfc_ is defined in the Lua module or is a variant of such a function; one which begins __lfc_ is defined in the package itself. This distinction disappears in the generated package code.

<*sty> <@@=lfc>

```
\NeedsTeXFormat{LaTeX2e}[2021-11-15]
\RequirePackage{svn-prov}
\ProvidesPackageSVN[\filebase.sty]{$Id: lua-font-config.dtx 12228 2026-11-24 22:01:15Z cfrees $}[v0.3 \revinfo][\filebase: Font configuration for LuaLaTeX]
% This is needed only because l3doc overwrites \fileversion and \filedate.
% Neither ltxdoc nor doc does this.
% But it seems better to do it this way on the basis of the general precautionary principle.
\DefineFileInfoSVN[lfc]

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% copied verbatim, excepting format from Joseph Wright's' siunitx.sty under LPPL
\@ifundefined{directlua}{%
  \PackageError{lua-font-config}{Package requires LuaLaTeX}
  {To use lua-font-config, you must compile your document with LuaLaTeX.}%
}{}
\@ifundefined{ExplLoaderFileDate}{%
  \RequirePackage{expl3}%
}{}
%% almost verbatim from siunitx.sty
\@ifl@t@r\ExplLoaderFileDate{2024-01-10}{%
}{%
  \PackageError{lua-font-config}{Support package expl3 too old}
  {%
    You need to update your installation of the bundles 'l3kernel' and
    'l3packages'.\MessageBreak
    Loading lua-font-config will abort!%
  }%
  \endinput
}
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
\ExplSyntaxOn
\prop_gput:NnV \g_msg_module_name_prop {lfc} \lfc@base
```

Messages.

Note that \msg_* are not recognised in the indexing; \msg appears instead. This is a limitation of the doc package, as far as I know.

```
\msg_new:nnnn {lfc} {needs-luatex} {
  \msg_critical_text:n {lfc} :: ~requires~LuaLaTeX~\msg_line_context:
} { \msg_see_documentation_text:n {lfc} }

\sys_if_engine_luatex:F { \msg_critical:nn {lfc} {needs-luatex} }

\msg_new:nnnn {lfc} {math-needs-lum} {
  #1~requires~lua-unicode-math,~but~
  you've~loaded~unicode-math.~This~cannot~work~\msg_line_context:.
} { To~configure~Unicode~maths~fonts~with~this~package,~we~need~to~load~
  lua-unicode-math,~but~unicode-math,~which~is~incompatible~has~already~
  been~loaded,~so~I'm~going~to~ignore~this~part~of~your~settings.~
```


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

```
  \msg_see_documentation_text:n {lfc}
}
\msg_new:nnnn {lfc} {loading} {
  Loading~#1~for~maths~support~\msg_line_context:.
} { \msg_see_documentation_text:n {lfc} }
\msg_new:nnnn {lfc} {meta-fam-missing} {
  Font~set~#1~not~defined.~Reported~\msg_line_context:.
} {
  A~set~of~fonts~(meta-family)~must~be~defined~in~the~preamble~before~it~can~
  be~used~in~your~document.~\msg_see_documentation_text:n {lfc}
}
\msg_new:nnnn {lfc} {meta-fam-exists} {
  Font~set~#1~already~defined.~Reported~\msg_line_context:.
} {
  You~tried~to~define~a~new~font~set~(meta-family)~with~the~same~name~as~an~
  existing~set.~\msg_see_documentation_text:n {lfc}
}
\msg_new:nnnn {lfc} {meta-fam-default-name} {
  Font~set~'default'~does~not~need~to~be~created.~
  Ignoring~request~\msg_line_context:.
} {
  You~tried~to~define~a~new~font~set~named~'default'~but~this~is~unneeded.~
  Instead,~use~\token_to_str:N \fontconfig ~to~configure~the~set.~
  \msg_see_documentation_text:n {lfc}
}
\msg_new:nnnn {lfc} {meta-fam-no-default} {
  The~'default'~font~set~is~not~configured.~Reported~\msg_line_context:.
} {
  You~tried~to~define~a~new~font~set~before~configuring~a~default~set.~
  This~is~not~possible.~You~need~first~to~use~\token_to_str:N \fontconfig ~
  to~configure~the~default~set.~\msg_see_documentation_text:n {lfc}
}
\msg_new:nnnn {lfc} {base-set-not-configured} {
  The~'#1'~font~set~is~not~configured.~Reported~\msg_line_context:.
} {
  You~tried~to~define~a~new~font~set~`#2'~based~on~existing~font~set~`#1',~
  before~configuring~`#1'.~This~is~not~possible.~You~need~first~to~use~
  \token_to_str:N \fontconfig [#1]~to~configure~the~base~set.~
  \msg_see_documentation_text:n {lfc}
}
\msg_new:nnn {lfc} {info} {
  #1.~
  Reported~\msg_line_context:.~
  #2.
}
```

Custom ephemeral variables, named variables and constants.

\l__lfc_features_seq (*var.*)
\l__lfc_scope_seq (*var.*)
\l__lfc_do_bool (*var.*)
\l__lfc_tmpa_tl (*var.*)
\l__lfc_tmpb_tl (*var.*)
\l__lfc_tmpa_str (*var.*)
\l__lfc_scope_str (*var.*)
\c__lfc_scope_default_str (*var.*)
\g__lfc_lum_done_bool (*var.*)
\l__lfc_tmpa_bool (*var.*)

```
\bool_new:N \l__lfc_do_bool
\bool_new:N \g__lfc_lum_done_bool
\bool_gset_false:N \g__lfc_lum_done_bool
\bool_new:N \l__lfc_tmpa_bool
\seq_new:N \l__lfc_features_seq
\seq_new:N \l__lfc_scope_seq
\str_new:N \l__lfc_scope_str
\str_new:N \l__lfc_tmpa_str
\str_const:Ne \c__lfc_scope_default_str {*\c_colon_str*}
\tl_new:N \l__lfc_tmpa_tl
\tl_new:N \l__lfc_tmpb_tl
```

`rm` (*key*)
`sf` (*key*)
`tt` (*key*)

Top-level keys for default text families.

```
\clist_map_inline:nn {rm, sf, tt} {
  \keys_define:nn {lfc} {
    #1 .code:n            = {
      \tl_clear:N \l__lfc_tmpa_tl
      \tl_clear:N \l__lfc_tmpb_tl
      \keys_set_exclude_groups:nnnN {lfc/#1} {config} {##1} \l__lfc_tmpa_tl
      \keys_set_groups:nnVN {lfc/#1} {config}     \l__lfc_tmpa_tl \l__lfc_tmpb_tl
      \keys_set_groups:nnV  {lfc/#1} {lateconfig} \l__lfc_tmpb_tl
      \bool_set_true:N \l__lfc_do_bool
    },
    #1 .value_required:n  = true,
    #1 .usage:n           = preamble,
  }
```

`.../name` (*key*)
`.../bfseries` (*key*)
`.../bold` (*key*)
`.../mdseries` (*key*)
`.../normal` (*key*)
`.../fakebold` (*key*)
`.../fakeoblique` (*key*)
`.../fea` (*key*)
`.../force` (*key*)
`.../scale` (*key*)
`.../scoped` (*key*)
`.../maths` (*key*)
`.../math` (*key*)
`.../nomaths` (*key*)
`.../nomath` (*key*)
`.../hyphenchar` (*key*)

Configure keys for default text families. Most of these just forward their values to `lfc/fam`.

```
  \keys_define:nn {lfc/#1} {
    name        .code:n   = { \use:c {__lfc_set_#1:n} {##1} },
    bfseries    .code:n   = {
      \__lfc_parse_series:nn {bfseries} {##1}
    },
    bold        .code:n   = {
      \str_set_eq:NN \l__lfc_weight_str \c_empty_str
      \str_set_eq:NN \l__lfc_width_str \c_empty_str
      \keys_set:nn {lfc/#1/series} {##1}
      \__lfc_parse_series:nVV {bfseries} \l__lfc_weight_str \l__lfc_width_str
    },
    mdseries    .code:n   = {
      \__lfc_parse_series:nn {mdseries} {##1}
    },
    normal      .code:n   = {
      \str_set_eq:NN \l__lfc_weight_str \c_empty_str
      \str_set_eq:NN \l__lfc_width_str \c_empty_str
      \keys_set:nn {lfc/#1/series} {##1}
      \__lfc_parse_series:nVV {mdseries} \l__lfc_weight_str \l__lfc_width_str
    },
    series / weight .str_set:N = \l__lfc_weight_str,
    series / width  .str_set:N = \l__lfc_width_str,
    fakebold    .code:n   = { \keys_set:nn {lfc/fam} {fakebold    = {##1}} },
    fakeoblique .code:n   = { \keys_set:nn {lfc/fam} {fakeoblique = {##1}} },
    fea         .code:n   = { \keys_set:nn {lfc/fam} {fea         = {##1}} },
    features    .code:n   = { \keys_set:nn {lfc/fam} {features    = {##1}} },
    force       .code:n   = { \keys_set:nn {lfc/fam} {force       = {##1}} },
    hyphenchar  .code:n   = { \keys_set:nn {lfc/fam} {hyphenchar  = {##1}} },
    scale       .code:n   = { \keys_set:nn {lfc/fam} {scale       = {##1}} },
    scoped      .code:n   = { \keys_set:nn {lfc/fam} {scoped      =  ##1}  },
    maths       .code:n   = { \__lfc_set_fam_cfg:nn   {maths}{##1} },
    math        .meta:n   = { maths = {##1} },
    nomaths     .meta:n   = { maths = false },
    nomath      .meta:n   = { nomaths },
    unknown     .code:n   = {
      \exp_args:NnV \use:c {__lfc_set_#1:n} \l_keys_key_str
    },
    name          .value_required:n   = true,
    math          .default:n          = true,
    maths         .default:n          = true,
    nomath        .value_forbidden:n  = true,
    nomaths       .value_forbidden:n  = true,
    unknown       .value_forbidden:n  = true,
```

```
    fea           .groups:n           = config,
    features      .groups:n           = lateconfig,
    force         .groups:n           = config,
    hyphenchar    .groups:n           = config,
    maths         .groups:n           = config,
    math          .groups:n           = config,
    nomath        .groups:n           = config,
    nomaths       .groups:n           = config,
    scale         .groups:n           = config,
    scoped        .groups:n           = config,
  }
}
```

debug (*key*)
math (*key*)
family (*key*)

Top-level keys for additional families, Unicode maths and debugging.

```
\keys_define:nn {lfc} {
  debug   .code:n           = {
    \tl_clear:N \l__lfc_tmpa_tl
    \keys_set_groups:nnnN
      {lfc/debug} {toggle} {state = true, #1} \l__lfc_tmpa_tl
    \keys_set:nV {lfc/debug} \l__lfc_tmpa_tl
  },
  debug   .usage:n          = general,
  family  .code:n           = {
    \tl_clear:N \l__lfc_tmpa_tl
    \tl_clear:N \l__lfc_tmpb_tl
    \str_clear:N \l__lfc_scope_str
    \keys_set_exclude_groups:nnnN
                          {lfc/fam} {config,lateconfig} {#1} \l__lfc_tmpa_tl
    \keys_set_groups:nnVN {lfc/fam} {config}      \l__lfc_tmpa_tl \l__lfc_tmpb_tl
    \keys_set_groups:nnV  {lfc/fam} {lateconfig}  \l__lfc_tmpb_tl
    \bool_set_true:N \l__lfc_do_bool
  },
  family  .value_required:n = true,
  family  .usage:n          = general,
  math    .code:n           = {
    \bool_if:NF \g__lfc_lum_done_bool {
      \IfPackageLoadedTF {unicode-math} {
        \msg_error:nnn {lfc} {math-needs-lum} {math config}
      } {
        \msg_info:nnn {lfc} {loading} {lua-unicode-math}
        \RequirePackage{lua-unicode-math}
        \bool_gset_true:N \g__lfc_lum_done_bool
      }
    }
    \tl_clear:N \l__lfc_tmpa_tl
    \keys_set_exclude_groups:nnnN {lfc/math} {config} {#1} \l__lfc_tmpa_tl
    \keys_set:nV {lfc/math} \l__lfc_tmpa_tl
    \bool_set_true:N \l__lfc_do_bool
  },
  math    .value_required:n = true,
  math    .usage:n          = preamble,
}
```

fam (*key*)

Configure keys for families.

fam/name (*key*)
fam/fakebold (*key*)
fam/fakeoblique (*key*)
fam/fea (*key*)
fam/features (*key*)
fam/force (*key*)
fam/hyphenchar (*key*)
fam/scale (*key*)
fam/scoped (*key*)

Configure keys for families.

Note to self: l3keys does not work the way you think!!! *The hierarchy is not a tree!*

```
\keys_define:nn {lfc/fam} {
```

```
name        .code:n = { \__lfc_set_fam_name:n {#1} },
command     .code:n = {
  \exp_args:Nne \__lfc_set_fam_cfg:nn {csname}{ \cs_to_str:N #1 }
},
fakebold    .code:n = {
  \__lfc_set_fam_cfg:nn {fakebold} {#1}
},
fakeoblique .code:n = {
  \__lfc_set_fam_cfg:nn {fakeoblique} {#1}
},
fea         .code:n = { \__lfc_set_fam_cfg:nn {fea} {#1} },
features    .code:n = {
  \seq_clear:N \l__lfc_features_seq
  \str_set:Ne \l__lfc_scope_str {*\c_colon_str*}

  \keys_set_groups:nnnN {lfc/features} {scope} {#1} \l__lfc_tempc_tl
  \keys_set:nV {lfc/features} \l__lfc_tempc_tl

  \seq_if_empty:NF \l__lfc_features_seq {
    \str_if_eq:eeTF {\l__lfc_scope_str} {\c__lfc_scope_default_str} {
      \exp_args:Nne \__lfc_set_fam_cfg:nn {fea} {
        \seq_use:Nn \l__lfc_features_seq {;}
      }
    } {
      \__lfc_set_fam_scoped_fea:Ve \l__lfc_scope_str {
        \seq_use:Nn \l__lfc_features_seq {;}
      }
    }
  }
  \str_clear:N \l__lfc_scope_str
},
force         .code:n = { \__lfc_set_fam_cfg:nn {force}{#1} },
hyphenchar    .choices:nn = {false,true}{
  \__lfc_set_fam_cfg:nn {hyphenchar}{#1}
},
scale         .code:n = { \__lfc_set_fam_cfg:nn {scale}{#1} },
scoped.code:n = {
  \str_set:Ne \l__lfc_tmpa_str {
    \__lfc_second_of_two:w #1 \q_stop
  }
  \str_if_empty:NTF \l__lfc_tmpa_str {
    \__lfc_set_fam_cfg:nn {fea} {#1}
  } {
    \__lfc_set_fam_scoped_fea:nn #1
  }
},
unknown   .code:n = {
  \exp_args:NV \__lfc_set_fam_name:n \l_keys_key_str
},
name          .value_required:n   = true,
fakebold      .default:n          = 1.5,
fakeoblique   .default:n          = 0.2,
fea           .value_required:n   = true,
features      .value_required:n   = true,
force         .default:n          = true,
hyphenchar    .default:n          = true,
scale         .value_required:n   = true,
scoped        .value_required:n   = true,
unknown       .value_forbidden:n  = true,
name          .usage:n            = general,
command       .usage:n            = general,
```

```
  fea           .usage:n           = general,
  scoped        .usage:n           = general,
  features      .usage:n           = general,
  force         .usage:n           = general,
  scale         .usage:n           = general,
  unknown       .usage:n           = general,
  command       .groups:n          = config,
  fakebold      .groups:n          = config,
  fakeoblique   .groups:n          = config,
  fea           .groups:n          = config,
  features      .groups:n          = lateconfig,
  force         .groups:n          = config,
  hyphenchar    .groups:n          = config,
  scale         .groups:n          = config,
  scoped        .groups:n          = config,
}
```

math/name (*key*)
math/force (*key*)
math/scale (*key*)

Configure keys for Unicode maths.

```
\IfPackageLoadedF {unicode-math} {
  \keys_define:nn {lfc/math} {
    name      .code:n = { \use:c {__lfc_set_math:n} {#1} },
    force     .code:n = { \keys_set:nn {lfc/fam} {force    = {#1}} },
    scale     .code:n = { \keys_set:nn {lfc/fam} {scale    = {#1}} },
    unknown   .code:n = {
      \exp_args:NnV \use:c {__lfc_set_math:n} \l_keys_key_str
    },
    name      .value_required:n   = true,
    force     .default:n          = true,
    scale     .value_required:n   = true,
    unknown   .value_forbidden:n  = true,
    name      .usage:n            = preamble,
    force     .usage:n            = preamble,
    scale     .usage:n            = preamble,
    unknown   .usage:n            = preamble,
    force     .groups:n           = config,
    scale     .groups:n           = config,
  }
}
```

features (*key*)
features/alternates (*key*)
features/embolden (*key*)
features/figures (*key*)
features/forms (*key*)
features/language (*key*)
features/ligatures (*key*)
features/mode (*key*)
features/scope (*key*)
features/script (*key*)
features/spacing (*key*)

Trying to be a more user-friendly wrapper around `fam/fea`.

```
\keys_define:nn {lfc/features} {
  alternates  .code:n = {
    \exp_args:NV \__lfc_set_fam_features:nnn \l__lfc_scope_str {alternates} {#1}
  },
  embolden    .code:n = {
    \seq_put_right:Nn \l__lfc_features_seq {embolden=#1}
  },
  figures     .code:n = {
    \exp_args:NV \__lfc_set_fam_features:nnn \l__lfc_scope_str {figures} {#1}
  },
  forms         .code:n = {
    \exp_args:NV \__lfc_set_fam_features:nnn \l__lfc_scope_str {forms} {#1}
  },
  language      .code:n = {
    \seq_put_right:Nn \l__lfc_features_seq {language=#1}
  },
  ligatures     .code:n = {
    \exp_args:NV \__lfc_set_fam_features:nnn \l__lfc_scope_str {ligatures} {#1}
  },
```

```
mode          .choices:nn = {base,node,harf}{
  \seq_put_right:Ne \l__lfc_features_seq {mode=\l_keys_choice_str}
},
scope           .code:n = {
  \seq_set_split:Nnn \l__lfc_scope_seq {,} {,,,}
  \keys_set:nn {lfc/features/scoped} {#1}
  \seq_map_indexed_inline:Nn \l__lfc_scope_seq {
    \int_case:nn {##1} {
      {1} { \str_set:Nn \l__lfc_tmpa_str {##2} }
      {2} {
        \bool_case:nF {
          { %% weight empty, width empty --> use *
            \bool_lazy_and_p:nn { \str_if_empty_p:n {##2} } {
              \str_if_empty_p:N \l__lfc_tmpa_str }
          } { \str_set:Nn \l__lfc_scope_str {*} }
          { %% weight m, width non-empty --> use width
            \bool_lazy_and_p:nn { \str_if_eq_p:Vn \l__lfc_tmpa_str {m} } {
              ! \str_if_empty_p:n {##2} }
          } { \str_set:Nn \l__lfc_scope_str {##2} }
          { %% weight non-empty, width m --> use weight
            \bool_lazy_and_p:nn { ! \str_if_empty_p:N \l__lfc_scope_str } {
              \str_if_eq_p:nn {##2} {m} }
          } { \str_set_eq:NN \l__lfc_scope_str \l__lfc_tmpa_str }
        } { %% o/w --> use concat
          \str_set:Ne \l__lfc_scope_str {\l__lfc_tmpa_str ##2}
        }
        \str_put_right:NV \l__lfc_scope_str \c_colon_str
      }
      {3} { \str_set:Nn \l__lfc_tmpa_str {##2} }
      {4} {
        \bool_case:nF {
          { %% shape empty, case empty --> use *
            \bool_lazy_and_p:nn { \str_if_empty_p:n {##2} } {
              \str_if_empty_p:N \l__lfc_tmpa_str
            }
          } { \str_put_right:Nn \l__lfc_scope_str {*} }
          { %% shape n, case non-empty --> use case
            \bool_lazy_and_p:nn { \str_if_eq_p:Vn \l__lfc_tmpa_str {n} } {
              ! \str_if_empty_p:n {##2} }
          } { \str_put_right:Nn \l__lfc_scope_str {##2} }
          { %% shape non-empty, case n --> use shape
            \bool_lazy_and_p:nn { ! \str_if_empty_p:N \l__lfc_tmpa_str } {
              \str_if_eq_p:nn {##2} {n} }
          } { \str_put_right:NV \l__lfc_scope_str \l__lfc_tmpa_str }
        } { %% o/w --> concat
          \str_put_right:Ne \l__lfc_scope_str {\l__lfc_tmpa_str ##2}
        }
      }
    }
  }
},
scoped / weight .choices:nn = {bold, book, demibold, extralight, extrabold,
  light, medium, regular, semibold, semilight, ultralight, ultrabold} {
  \exp_args:NNne \seq_set_item:Nnn \l__lfc_scope_seq {1} {
    \__lfc_parse_spec:nV {weight} \l_keys_choice_str
  }
},
scoped / width  .choices:nn = {condensed, expanded, extracondensed,
  extraexpanded, medium, semicondensed, semiexpanded, ultracondensed,
  ultraexpanded} {
  \exp_args:NNne \seq_set_item:Nnn \l__lfc_scope_seq {2} {
```

```
      \__lfc_parse_spec:nV {width} \l_keys_choice_str
    }
  },
  scoped / case   .choices:nn = {mixed, smallcaps} {
    \exp_args:NNne \seq_set_item:Nnn \l__lfc_scope_seq {3} {
      \__lfc_parse_spec:nV {case} \l_keys_choice_str
    }
  },
  scoped / style  .choices:nn = {italic, oblique, upright} {
    \exp_args:NNne \seq_set_item:Nnn \l__lfc_scope_seq {4} {
      \__lfc_parse_spec:nV {shape} \l_keys_choice_str
    }
  },
  script        .code:n = {
    \seq_put_right:Nn \l__lfc_features_seq {script=#1}
  },
  spacing       .code:n = {
    \exp_args:NV \__lfc_set_fam_features:nnn \l__lfc_scope_str {spacings} {#1}
  },
  alternates  .value_required:n = true,
  embolden    .value_required:n = true,
  figures     .value_required:n = true,
  forms       .value_required:n = true,
  language    .value_required:n = true,
  ligatures   .value_required:n = true,
  mode        .value_required:n = true,
  scope       .value_required:n = true,
  script      .value_required:n = true,
  spacing     .value_required:n = true,
  scope       .groups:n         = scope,
}
```

debug/state (*key*)
debug/categories (*key*)

Configure keys for debugging.

```
\keys_define:nn {lfc/debug} {
  categories  .code:n            = { \__lfc_debug_set_cats_scanner:n {#1} },
  state       .choice:,
  state/false .code:n            = { \__lfc_debug_set_false: },
  state/true  .code:n            = { \__lfc_debug_set_true: },
  categories  .usage:n           = general,
  state       .usage:n           = general,
  categories  .value_required:n  = true,
  state       .default:n         = true,
  state       .groups:n          = toggle,
}
```

__lfc_font_config:nn (*fn.*)

The same internal function supports standard and multiple configuration macros.

```
\cs_new_protected:Npn \__lfc_font_config:nn #1#2 {
  \bool_set_false:N \l__lfc_do_bool
  \__lfc_set_font_set:n {#1}
  \keys_set:nn {lfc} {#2}
  \bool_if:NT \l__lfc_do_bool {
    \__lfc_configure_doc_families:
  }
}
```

\fontconfig

The main configuration function and main public macro.

```
\NewDocumentCommand\fontconfig { O {default} m } {
  \__lfc_font_config:nn {#1} {#2}
```

```
}
```

\newfontset

```
\NewDocumentCommand\newfontset { o m } {
  \str_if_eq:eeTF {default} {#2} {
    \msg_warning:nn {lfc} {meta-fam-default-name}
  } {
    \cs_if_exist:NF \__lfc_font_set_default:  {
      \msg_error:nn {lfc} {meta-fam-no-default}
    }
    \bool_if:NF \g__lfc_lum_done_bool {
      \IfPackageLoadedTF {unicode-math} {
        \msg_error:nnn {lfc} {math-needs-lum} {font sets}
      } {
        \msg_info:nnn {lfc} {loading} {lua-unicode-math}
        \RequirePackage{lua-unicode-math}
        \bool_gset_true:N \g__lfc_lum_done_bool
      }
    }
    \cs_if_free:cTF { __lfc_font_set_#2: } {
      \tl_if_novalue:nTF {#1} {
        \__lfc_new_font_set:n {#2}
      } {
        \cs_if_free:cTF { __lfc_font_set_#1: } {
          \msg_error:nnnn {lfc} {base-set-not-configured} {#1} {#2}
        } {
          \__lfc_new_font_set:nn {#1} {#2}
        }
      }
    } {
      \msg_error:nnn {lfc} {meta-fam-exists} {#2}
    }
  }
}
```

\usefontset \usefontset is a simple wrapper around the Lua functions.

```
\NewDocumentCommand\usefontset { m } {
  \cs_if_exist_use:cF { __lfc_font_set_#1: } {
    \msg_error:nnn {lfc} {meta-fam-missing} {#1}
  }
}
```

\normalmath The antithesis of \boldmath.

```
\cs_new:Npe \normalmath {
  \exp_not:N \protect \exp_not:c {normalmath\c_space_tl}
}
\cs_new:cpn {normalmath\c_space_tl} {
  \@nomath\normalmath \mathversion {normal}
}
```

__lfc_set_mathrm:n (*fn.*)
__lfc_set_mathsf:n (*fn.*)
__lfc_set_mathtt:n (*fn.*)
__lfc_set_mathrm: (*fn.*)
__lfc_set_mathsf: (*fn.*)
__lfc_set_mathtt: (*fn.*)
__lfc_set_mathrm:nnnnnn (*fn.*)
__lfc_set_mathsf:nnnnn (*fn.*)
__lfc_set_mathtt:nnnnn (*fn.*)
__lfc_set_mathrm:nnVVV (*fn.*)
__lfc_set_mathsf:nnVVV (*fn.*)
__lfc_set_mathtt:nnVVV (*fn.*)

These are set up slightly differently for default and non-default font sets.

```
\cs_new_protected_nopar:Npn \__lfc_set_mathrm:nnnnnn #1#2#3#4#5#6 {
  \tl_if_blank:nTF {#1} {
    \SetMathAlphabet \mathrm {normal} {TU} {#2} {#3} {#5}
    \SetMathAlphabet \mathit {normal} {TU} {#2} {#3} {#6}
    \SetMathAlphabet \mathbf {normal} {TU} {#2} {#4} {#5}
```

```
    \SetMathAlphabet \mathrm {bold}   {TU} {#2} {#4} {#5}
    \SetMathAlphabet \mathit {bold}   {TU} {#2} {#4} {#6}
    \SetMathAlphabet \mathbf {bold}   {TU} {#2} {#4} {#5}
    \SetSymbolFont{operators}{normal} {TU} {#2} {#3} {#5}
    \SetSymbolFont{operators}{bold}   {TU} {#2} {#4} {#5}
    \cs_gset_protected_nopar:Npn \__lfc_set_mathrm: {}
  } {
    \SetMathAlphabet \mathrm {normal@#1} {TU} {#2} {#3} {#5}
    \SetMathAlphabet \mathit {normal@#1} {TU} {#2} {#3} {#6}
    \SetMathAlphabet \mathbf {normal@#1} {TU} {#2} {#4} {#5}
    \SetMathAlphabet \mathrm {bold@#1}   {TU} {#2} {#4} {#5}
    \SetMathAlphabet \mathit {bold@#1}   {TU} {#2} {#4} {#6}
    \SetMathAlphabet \mathbf {bold@#1}   {TU} {#2} {#4} {#5}
    \SetSymbolFont{operators}{normal@#1} {TU} {#2} {#3} {#5}
    \SetSymbolFont{operators}{bold@#1}   {TU} {#2} {#4} {#5}
  }
}
\cs_new_protected_nopar:Npn \__lfc_set_mathsf:nnnnn #1#2#3#4#5 {
  \tl_if_blank:nTF {#1} {
    \SetMathAlphabet \mathsf {normal} {TU} {#2} {#3} {#5}
    \SetMathAlphabet \mathsf {bold}   {TU} {#2} {#4} {#5}
    \cs_gset_protected_nopar:Npn \__lfc_set_mathsf: {}
  } {
    \SetMathAlphabet \mathsf {normal@#1} {TU} {#2} {#3} {#5}
    \SetMathAlphabet \mathsf {bold@#1}   {TU} {#2} {#4} {#5}
  }
}
\cs_new_protected_nopar:Npn \__lfc_set_mathtt:nnnnn #1#2#3#4#5 {
  \tl_if_blank:nTF {#1} {
    \SetMathAlphabet \mathtt {normal} {TU} {#2} {#3} {#5}
    \SetMathAlphabet \mathtt {bold}   {TU} {#2} {#4} {#5}
    \cs_gset_protected_nopar:Npn \__lfc_set_mathtt: {}
  } {
    \SetMathAlphabet \mathtt {normal@#1} {TU} {#2} {#3} {#5}
    \SetMathAlphabet \mathtt {bold@#1}   {TU} {#2} {#4} {#5}
  }
}
\cs_generate_variant:Nn \__lfc_set_mathrm:nnnnnn {nnVVVV}
\cs_generate_variant:Nn \__lfc_set_mathsf:nnnnn {nnVVV}
\cs_generate_variant:Nn \__lfc_set_mathtt:nnnnn {nnVVV}
\cs_new_protected_nopar:Npn \__lfc_set_mathrm:n #1 {
  \__lfc_set_mathrm:nnVVVV  {} {#1} \mdseries@rm \bfseries@rm \shapedefault
    \itdefault
}
\cs_new_protected_nopar:Npn \__lfc_set_mathsf:n #1 {
  \__lfc_set_mathsf:nnVVV  {} {#1} \mdseries@sf \bfseries@sf \shapedefault
}
\cs_new_protected_nopar:Npn \__lfc_set_mathtt:n #1 {
  \__lfc_set_mathtt:nnVVV  {} {#1} \mdseries@tt \bfseries@tt \shapedefault
}
\cs_new_protected_nopar:Npn \__lfc_set_mathrm: {
  \SetMathAlphabet      \mathrm {normal} {TU} {lmr-base}  {m}   {n}
  \SetMathAlphabet      \mathit {normal} {TU} {lmr-base}  {m}   {it}
  \SetMathAlphabet      \mathbf {normal} {TU} {lmr-base}  {bx}  {n}
  \SetMathAlphabet      \mathrm {bold}   {TU} {lmr-base}  {bx}  {n}
  \SetMathAlphabet      \mathit {bold}   {TU} {lmr-base}  {bx}  {it}
  \SetMathAlphabet      \mathbf {bold}   {TU} {lmr-base}  {bx}  {n}
  \DeclareSymbolFont{operators}          {TU} {lmr-base}  {m}   {n}
  \SetSymbolFont    {operators} {bold}   {TU} {lmr-base}  {bx}  {n}
}
\cs_new_protected_nopar:Npn \__lfc_set_mathsf: {
```

```
  \SetMathAlphabet      \mathsf {normal} {TU} {lmss-base} {m}   {n}
  \SetMathAlphabet      \mathsf {bold}   {TU} {lmss-base} {bx}  {n}
}
\cs_new_protected_nopar:Npn \__lfc_set_mathtt: {
  \SetMathAlphabet      \mathtt {normal} {TU} {lmtt-base} {m}   {n}
  \SetMathAlphabet      \mathtt {bold}   {TU} {lmtt-base} {bx}  {n}
}
```

`eclare_lum_symbol_fonts:nn` (*fn.*)
`_set_lum_symbol_fonts:nnn` (*fn.*)

Maths setup functions.

```
\cs_new_protected_nopar:Npn \__lfc_declare_lum_symbol_fonts:nn #1#2 {
  \DeclareSymbolFont{lummain}{TU}{#1}{m}{n}
  \SetSymbolFont{lummain}{bold}{TU}{#2}{b}{n}
}
\cs_new_protected_nopar:Npn \__lfc_set_lum_symbol_fonts:nnnn #1#2#3#4 {
  \tl_if_blank:nTF {#1} {
    \SetSymbolFont{lummain}{normal}   {TU}{#2}{#3}{n}
    \SetSymbolFont{lummain}{bold}     {TU}{#2}{#4}{n}
  } {
    \SetSymbolFont{lummain}{normal@#1}{TU}{#2}{#3}{n}
    \SetSymbolFont{lummain}{bold@#1}  {TU}{#2}{#4}{n}
  }
}
```

`\__lfc_second_of_two:w` (*fn.*)

Auxiliary because TEX still makes more sense to me here. I know I'm meant to use something like `\use_romannumeral_delimited_by_something:w` but figuring out which …. (I can just search for the translation of 'firstof', but there isn't one to find for 'secondof' ….)

```
\cs_new_nopar:Npn \__lfc_second_of_two:w #1#2 \q_stop {#2}
```

`\__lfc_no_hyphenchar:` (*fn.*)

Zap hyphen character. A more general approach should be used here. I don't *think* this should be protected?

```
\cs_new_nopar:Npn \__lfc_no_hyphenchar: {\hyphenchar\font\m@ne}
```

Compatibility code for ltx-talk.

This should be equivalent to the default setup, modulo any differences between emulating the package and using `\setmathfont` for lua-unicode-math.

```
\hook_gput_code:nnn {class/ltx-talk/before} {.} {
  \hook_gput_code:nnn {package/mathtools/after} {.} {
    \RequirePackage{lua-unicode-math}
    \fontconfig{
      rm      = {newcomputermodernsans},
      sf      = {newcomputermodernsans},
      math    = {newcomputermodernsansmath},
    }
    \tagpdfsetup { math / mathml / luamml / load = true }
  }
}
```

If lua-unicode-math is loaded (either by us or otherwise), set default math alphabets to match the default math font. This is not perfect, but hopefully reasonably safe[18]. This may go wrong if the maths alphabets are set earlier by alternative means e.g. if fontspec is loaded and used to set one or more of the main document families. I *think* lua-unicode-math will override our settings in this case, so it should be all right, maybe?

[18] Hoping it is not necessary to do additional checks here?

```
\hook_gput_code:nnn {package/lua-unicode-math/after} {.} {
  \__lfc_set_mathrm:
  \__lfc_set_mathsf:
  \__lfc_set_mathtt:
}
```

The Lua modules do the actual work.

```
\lua_load_module:n {lua-font-config}
\cs_generate_variant:Nn \__lfc_set_fam_scoped_fea:nn {Ve}
\cs_generate_variant:Nn \__lfc_parse_spec:nn {nV}
\cs_generate_variant:Nn \__lfc_parse_series:nnn {nVV}
```

__lfc_set_fam_cfg:nn (*fn.*)
__lfc_set_fam_fea_scoped:nn (*fn.*)
__lfc_set_fam_fea_scoped:Ve (*fn.*)
__lfc_set_fam_features:nnn (*fn.*)
__lfc_set_fam_name:n (*fn.*)
__lfc_set_font_set:n (*fn.*)
__lfc_new_font_set:n (*fn.*)
__lfc_new_font_set:nn (*fn.*)
__lfc_parse_series:nn (*fn.*)
__lfc_parse_series:nnn (*fn.*)
__lfc_parse_series:nVV (*fn.*)
__lfc_parse_spec:nn (*fn.*)
__lfc_parse_spec:nV (*fn.*)
__lfc_set_mathrm: (*fn.*)
__lfc_set_mathsf: (*fn.*)
__lfc_set_mathtt: (*fn.*)
__lfc_configure_doc_families: (*fn.*)
__lfc_debug_set_cats_scanner:n (*fn.*)
__lfc_debug_set_false: (*fn.*)
__lfc_debug_set_true: (*fn.*)

Font sets get functios such as `\__lfc_font_set_default:` where `default` is either literal or the name of a configured meta-family.

The following removes various commands from `\@preamblecmds`, allowing them to be used in the body of the document and not only in the preamble.

This approach is based on David Carlisle's suggestions at https://chat.stackexchange.com/transcript/message/69314513#69314513. David suggested setting the entire list `\@preamblecmds` empty, but I don't really think an alternative font loader (?) should be in the business of e.g. enabling use of `\usepackage` (or `\documentstyle`!) inside the `document` environment. So let's try editing the list instead ….

We don't enable anything disallowed for text fonts. But we do want to allow new math versions and the assignment of maths fonts to those versions: specifically, we want `\SetMathAlphabet` and `\SetSymbolFont` to be usable, as well as configuration of new Unicode math fonts. This avoids the need to predefine additional maths versions for switching font sets, so we can define them only if the font set is actually enabled in the document. It also means we can avoid even doing the configuration and setting the definitions up if a set is never used.

\@preamblecmds Listed macros[19] will be permitted in the document, overriding the LaTeX restriction.

```
\tl_set:Nn \l__lfc_tmpb_tl {
  \do
  \alpha@elt
  \alpha@list
  \cdp@elt
  \cdp@list
  \DeclareMathVersion
  \DeclareMathAlphabet
  \DeclareSymbolFont
  \dorestore@version
  \get@cdp
  \group@elt
  \group@list
  \init@restore@version
  \new@mathalphabet
  \new@mathgroup
  \new@mathversion
  \new@symbolfont
  \restore@mathversion
  \SetMathAlphabet
  \SetMathAlphabet@
  \SetSymbolFont@
  \SetSymbolFont
  \version@elt
  \version@list
```

[19] Except `do`, obviously.

```
}
\tl_clear:N \l__lfc_tmpa_tl
\tl_map_inline:Nn \@preamblecmds {
  \bool_set_true:N \l__lfc_tmpa_bool
  \tl_map_inline:Nn \l__lfc_tmpb_tl {
    \tl_if_eq:nnT {##1} {#1} {
      \bool_set_false:N \l__lfc_tmpa_bool
      \tl_map_break:
    }
  }
  \bool_if:NT \l__lfc_tmpa_bool {
    \tl_put_right:Nn \l__lfc_tmpa_tl {\do #1}
  }
}
\tl_set_eq:NN \@preamblecmds \l__lfc_tmpa_tl
```

We are hypocrites: we help make LaTeX preamble-only macros available in the document body, but we disallow use of our own there. This restriction is probably not necessary, but it is imposed pending the discovery of some compelling use-case requiring use in the document body.

For now, then, `\newfontset` and `\fontconfig` should not be used inside `document`. Is there a (similarly simple) way to do this in expl3? Well, I just learnt this from babel …, so maybe I'll find it in another few decades or so.

```
\ExplSyntaxOff
\@onlypreamble \newfontset
\@onlypreamble \fontconfig
```

`\UTFencname` Only if it doesn't exist. I'm not sure whether this is actually useful (yet?) ….

```
\ifcsname UTFencname\endcsname\relax\else\def\UTFencname{TU}\fi
```

</sty>

Change History

v0.1
  General: First public release. 25
v0.2
  General: Added \l__lfc_features_seq, \l__lfc_tmpb_tl. 27
    Adjusted to support addition of features interface. 29
    Bug fixes, code cleanup, somewhat better UI, doc improvements. 25
  fam: Add fea user-interface wrapper: features provides a key-value interface for configuration. 29
  features: Initial features keys: alternates, embolden, figures, forms, language, ligatures, node, script, spacing. 31
  \fontconfig: Forbid \par in input. 33
v0.3
  General: __lfc_set_fam_features:nnn replaces __lfc_set_fam_features:nn. Added __lfc_set_fam_fea_scoped:nn, __lfc_parse_spec:nn, , __lfc_parse_series:nn, __lfc_parse_series:nnn. 37
Accommodate multiple font sets. 36
Add __lfc_set_font_set:n. 37
Add key to disable hyphenation character. 28
Add keys to configure bold and regular series for default families and to enable use of fake bold and oblique. 28
Added \g__lfc_lum_done_bool. 27
Added \g__lfc_tmpa_bool. 27
Added temporary string, tl and \l__lfc_do_bool, \l__lfc_scope_str, \l__lfc_scope_seq, \c__lfc_scope_default_str. 27
Addition of disposable variants …. 34
Extended to support multiple font sets. 34
Introduction of font sets to support multiple switchable font configurations. Two new public commands: \newfontset and \usefontset allow users to manage these and \fontconfig gets a new optional argument to accommodate them. 25

Separate user documentation. 25
__lfc_font_config:nn: New internal function to support multiple configurations. 33
fam: Add scoped key: low-level interface to configure font features selectively. This is intended to complement the fea key and similarly supports expert/advanced usage, prioritising performance with few checks and a relatively low-level interface. 29
fam/scoped: Add hyphenchar. 29
Add scoped, fakebold, fakeoblique. . . 29
features: Add scope key. This provides a more user-friendly wrapper around the scoped key. 31
Renamed standard, nostandard ligatures keys common, nocommon as this seems more ..., er, standard. 31
\fontconfig: Optional argument to set a different 'meta-family'. 33
\newfontset: New macro to define a new font set. 34
\usefontset: New macro to switch font sets. 34

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.

Symbols

.../bfseries (key) 111
.../bold (key) 111
.../fakebold (key) 111
.../fakeoblique (key) 111
.../fea (key) 111
.../force (key) 111
.../hyphenchar (key) 111
.../math (key) 111
.../maths (key) 111
.../mdseries (key) 111
.../name (key) 111
.../nomath (key) 111
.../nomaths (key) 111
.../normal (key) 111
.../scale (key) 111
.../scoped (key) 111
\@nomath 475
\@onlypreamble 626, 627
\@preamblecmds 585
__lfc_configure_doc_families: (fn.) 429, 581
__lfc_debug_set_cats_scanner:n (fn.) 414, 581
__lfc_debug_set_false: (fn.) 416, 581
__lfc_debug_set_true: (fn.) 417, 581
__lfc_declare_lum_symbol_fonts:nn (fn.) 550
__lfc_font_config:nn (fn.) 424, 433
__lfc_font_set_default: 439
__lfc_new_font_set:n (fn.) 453, 581
__lfc_new_font_set:nn (fn.) 458, 581
__lfc_no_hyphenchar: (fn.) 564
__lfc_parse_series:nVV (fn.) . . 120, 129, 581
__lfc_parse_series:nn (fn.) . . . 114, 123, 581
__lfc_parse_series:nnn (fn.) 581
__lfc_parse_spec:nV (fn.) 375, 382, 387, 392, 581
__lfc_parse_spec:nn (fn.) 581
__lfc_second_of_two:w (fn.) 243, 563
__lfc_set_fam_cfg:nn (fn.) 141, 207, 210, 213, 215, 225, 236, 238, 240, 246, 581
__lfc_set_fam_fea_scoped:Ve (fn.) 581
__lfc_set_fam_fea_scoped:nn (fn.) 581
__lfc_set_fam_features:nnn (fn.) 304, 310, 313, 319, 399, 581
__lfc_set_fam_name:n (fn.) 205, 252, 581
__lfc_set_fam_scoped_fea:Ve 229
__lfc_set_fam_scoped_fea:nn 248, 582
__lfc_set_font_set:n (fn.) 426, 581
__lfc_set_lum_symbol_fonts:nnn (fn.) . . 550
__lfc_set_lum_symbol_fonts:nnnn 554
__lfc_set_mathrm: (fn.) 477, 577, 581
__lfc_set_mathrm:n (fn.) 477
__lfc_set_mathrm:nnVVV (fn.) 477
__lfc_set_mathrm:nnnnnn (fn.) 477
__lfc_set_mathsf: (fn.) 477, 578, 581
__lfc_set_mathsf:n (fn.) 477
__lfc_set_mathsf:nnVVV (fn.) 477
__lfc_set_mathsf:nnnnn (fn.) 477
__lfc_set_mathtt: (fn.) 477, 579, 581
__lfc_set_mathtt:n (fn.) 477
__lfc_set_mathtt:nnVVV (fn.) 477
__lfc_set_mathtt:nnnnn (fn.) 477

A

\alpha@elt 587
\alpha@list 588
alternates (key) *6*

B

bfseries (key) *9*
\bfseries@rm 523
\bfseries@sf 527
\bfseries@tt 530
bold (key) *9*
\boldmath *12*
\bool_case:nF 331, 351
\bool_gset_false:N 89
\bool_gset_true:N 193, 448
\bool_if:NF 187, 442
\bool_if:NT 428, 620
\bool_lazy_and_p:nn 333, 337, 341, 353, 358, 362
\bool_new:N 87, 88, 90

\bool_set_false:N 425, 616
\bool_set_true:N 106, 182, 199, 613

C

\c__lfc_scope_default_str (var.) 87, 224
\c_colon_str 95, 218, 347
\c_empty_str 117, 118, 126, 127
\c_space_tl 472, 474
case (key) 8
\cdp@elt 589
\cdp@list 590
\clist_map_inline:nn 98
command (key) 10
\cs_generate_variant:Nn .....
..... 519, 520, 521, 582, 583, 584
\cs_gset_protected_nopar:Npn . 487, 503, 513
\cs_if_exist:NF 439
\cs_if_exist_use:cF 467
\cs_if_free:cTF 451, 455
\cs_new:cpn 474
\cs_new:Npe 471
\cs_new_nopar:Npn 563, 564
\cs_new_protected:Npn 424
\cs_new_protected_nopar:Npn ... 477, 499,
509, 522, 526, 529, 532, 542, 546, 550, 554
\cs_to_str:N 207

D

debug (key) 3, 166
debug/categories (key) 413
debug/state (key) 413
\DeclareMathAlphabet 592
\DeclareMathVersion 591
\DeclareSymbolFont 539, 551, 593
\def 628
\do 586, 621
\dorestore@version 594

E

\else 628
embolden (key) 6
\endcsname 628
\exp_args:Nne 207, 225
\exp_args:NNne 374, 381, 386, 391
\exp_args:NnV 146, 288
\exp_args:NV 252, 304, 310, 313, 319, 399
\exp_not:c 472
\exp_not:N 472

F

fakebold (key) 8
fakeoblique (key) 8
fam (key) 204
fam/fakebold (key) 204
fam/fakeoblique (key) 204
fam/fea (key) 204
fam/features (key) 204
fam/force (key) 204
fam/hyphenchar (key) 204
fam/name (key) 204
fam/scale (key) 204
fam/scoped (key) 204
family (key) 3, 3, 166
fea (key) 12
features (key) 5, 302
features/alternates (key) 302
features/embolden (key) 302
features/figures (key) 302
features/forms (key) 302
features/language (key) 302
features/ligatures (key) 302
features/mode (key) 302
features/scope (key) 302
features/script (key) 302
features/spacing (key) 302
\fi 628
figures (key) 6
\font 564
\fontconfig 3, 64, 71, 79, 432, 568, 627
force (key) 5
forms (key) 6
functions:
__lfc_configure_doc_families: . 429, 581
__lfc_debug_set_cats_scanner:n 414, 581
__lfc_debug_set_false: 416, 581
__lfc_debug_set_true: 417, 581
__lfc_declare_lum_symbol_fonts:nn 550
__lfc_font_config:nn 424, 433
__lfc_new_font_set:n 453, 581
__lfc_new_font_set:nn 458, 581
__lfc_no_hyphenchar: 564
__lfc_parse_series:nVV ... 120, 129, 581
__lfc_parse_series:nn 114, 123, 581
__lfc_parse_series:nnn 581
__lfc_parse_spec:nV 375, 382, 387, 392, 581
__lfc_parse_spec:nn 581
__lfc_second_of_two:w 243, 563
__lfc_set_fam_cfg:nn 141, 207,
210, 213, 215, 225, 236, 238, 240, 246, 581
__lfc_set_fam_fea_scoped:Ve 581
__lfc_set_fam_fea_scoped:nn 581
__lfc_set_fam_features:nnn .....
..... 304, 310, 313, 319, 399, 581
__lfc_set_fam_name:n 205, 252, 581
__lfc_set_font_set:n 426, 581
__lfc_set_lum_symbol_fonts:nnn ... 550
__lfc_set_mathrm: 477, 577, 581
__lfc_set_mathrm:n 477
__lfc_set_mathrm:nnVVVV 477
__lfc_set_mathrm:nnnnnn 477
__lfc_set_mathsf: 477, 578, 581
__lfc_set_mathsf:n 477
__lfc_set_mathsf:nnVVV 477
__lfc_set_mathsf:nnnnn 477
__lfc_set_mathtt: 477, 579, 581
__lfc_set_mathtt:n 477
__lfc_set_mathtt:nnVVV 477
__lfc_set_mathtt:nnnnn 477

G

\g__lfc_lum_done_bool (var.) .....
..... 87, 187, 193, 442, 448
\g_msg_module_name_prop 31

\get@cdp 595
\group@elt 596
\group@list 597

H

\hook_gput_code:nnn 565, 566, 576
\hyphenchar 564
hyphenchar (key) *7*

I

\ifcsname 628
\IfPackageLoadedF 282
\IfPackageLoadedTF 188, 443
\init@restore@version 598
\int_case:nn 328
\itdefault 524

K

keys:
.../bfseries 111
.../bold 111
.../fakebold 111
.../fakeoblique 111
.../fea 111
.../force 111
.../hyphenchar 111
.../math 111
.../maths 111
.../mdseries 111
.../name 111
.../nomath 111
.../nomaths 111
.../normal 111
.../scale 111
.../scoped 111
alternates *6*
bfseries *9*
bold *9*
case *8*
command *10*
debug *3*, 166
debug/categories 413
debug/state 413
embolden *6*
fakebold *8*
fakeoblique *8*
fam 204
fam/fakebold 204
fam/fakeoblique 204
fam/fea 204
fam/features 204
fam/force 204
fam/hyphenchar 204
fam/name 204
fam/scale 204
fam/scoped 204
family *3*, *3*, 166
fea *12*
features *5*, 302
features/alternates 302
features/embolden 302
features/figures 302
features/forms 302
features/language 302
features/ligatures 302
features/mode 302
features/scope 302
features/script 302
features/spacing 302
figures *6*
force *5*
forms *6*
hyphenchar *7*
language *7*
ligatures *7*
math *3*, *3*, *9*, 166
math/force 282
math/name 282
math/scale 282
maths *9*
mdseries *9*
mode *7*
name *4*
nomath *9*
nomaths *9*
normal *9*
rm *3*, *3*, 98
scale *5*
scope *7*
scoped *13*
script *7*
sf *3*, *3*, 98
spacing *8*
style *8*
tt *3*, *3*, 98
weight *8*, *9*
width *8*, *9*
\keys_define:nn 99, 111, 166, 204, 283, 302, 413
\keys_set:nn 119, 128, 133, 134, 135, 136, 137, 138, 139, 140, 285, 286, 326, 427
\keys_set:nV 171, 198, 221
\keys_set_exclude_groups:nnnN . 103, 178, 197
\keys_set_groups:nnnN 169, 220
\keys_set_groups:nnV 105, 181
\keys_set_groups:nnVN 104, 180

L

\l__lfc_do_bool (var.) 87, 106, 182, 199, 425, 428
\l__lfc_features_seq (var.) 87, 217, 223, 226, 230, 307, 316, 322, 396
\l__lfc_scope_seq (var.) 87, 325, 327, 374, 381, 386, 391
\l__lfc_scope_str (var.) . 87, 177, 218, 224, 229, 234, 304, 310, 313, 319, 335, 339, 341, 343, 345, 347, 356, 360, 364, 366, 399
\l__lfc_tempc_tl 220, 221
\l__lfc_tmpa_bool (var.) 87, 613, 616, 620
\l__lfc_tmpa_str (var.) 87, 242, 245, 329, 334, 337, 343, 345, 349, 354, 358, 362, 364, 366
\l__lfc_tmpa_tl (var.) 87, 101, 103, 104, 168, 170, 171, 175, 179, 180, 196, 197, 198, 611, 621, 624

\l__lfc_tmpb_tl (var.) 87, 102, 104, 105, 176, 180, 181, 585, 614
\l__lfc_weight_str . . . 117, 120, 126, 129, 131
\l__lfc_width_str 118, 120, 127, 129, 132
\l_keys_choice_str . . . 322, 375, 382, 387, 392
\l_keys_key_str 146, 252, 288
language (key) 7
\lfc@base 31
ligatures (key) 7
\lua_load_module:n 581

M

\m@ne 564
math (key) 3, 3, 9, 166
math/force (key) 282
math/name (key) 282
math/scale (key) 282
\mathbf 481, 484, 491, 494, 535, 538
\mathit 480, 483, 490, 493, 534, 537
\mathrm 479, 482, 489, 492, 533, 536
maths (key) 9
\mathsf 501, 502, 505, 506, 543, 544
\mathtt 511, 512, 515, 516, 547, 548
\mathversion 475
mdseries (key) 9
\mdseries@rm 523
\mdseries@sf 527
\mdseries@tt 530
mode (key) 7
\msg_critical:nn 35
\msg_error:nn 440
\msg_error:nnn 189, 444, 462, 468
\msg_error:nnnn 456
\msg_info:nnn 191, 446
\msg_warning:nn 437

N

name (key) 4
\new@mathalphabet 599
\new@mathgroup 600
\new@mathversion 601
\new@symbolfont 602
\NewDocumentCommand 432, 435, 466
\newfontset 12, 435, 626
nomath (key) 9
nomaths (key) 9
normal (key) 9
\normalmath 12, 471

P

\prop_gput:NnV 31
\protect 472

Q

\q_stop 243, 563

R

\relax 628
\restore@mathversion 603
rm (key) 3, 3, 98

S

scale (key) 5
scope (key) 7
scoped (key) 13
script (key) 7
\seq_clear:N 217
\seq_if_empty:NF 223
\seq_map_indexed_inline:Nn 327
\seq_new:N 91, 92
\seq_put_right:Ne 322
\seq_put_right:Nn 307, 316, 396
\seq_set_item:Nnn 374, 381, 386, 391
\seq_set_split:Nnn 325
\seq_use:Nn 226, 230
\SetMathAlphabet 479, 480, 481, 482, 483, 484, 489, 490, 491, 492, 493, 494, 501, 502, 505, 506, 511, 512, 515, 516, 533, 534, 535, 536, 537, 538, 543, 544, 547, 548, 604
\SetMathAlphabet@ 605
\SetSymbolFont 485, 486, 495, 496, 540, 552, 556, 557, 559, 560, 607
\SetSymbolFont@ 606
sf (key) 3, 3, 98
\shapedefault 523, 527, 530
spacing (key) 8
\str_clear:N 177, 234
\str_const:Ne 95
\str_if_empty:NTF 245
\str_if_empty_p:N 334, 341, 354, 362
\str_if_empty_p:n 333, 338, 353, 359
\str_if_eq:eeTF 224, 436
\str_if_eq_p:nn 342, 363
\str_if_eq_p:Vn 337, 358
\str_new:N 93, 94
\str_put_right:Ne 366
\str_put_right:Nn 356, 360
\str_put_right:NV 347, 364
\str_set:Ne 218, 242, 345
\str_set:Nn 329, 335, 339, 349
\str_set_eq:NN 117, 118, 126, 127, 343
style (key) 8
\sys_if_engine_luatex:F 35

T

\tagpdfsetup 573
\tl_clear:N . . 101, 102, 168, 175, 176, 196, 611
\tl_if_blank:nTF 478, 500, 510, 555
\tl_if_eq:nnT 615
\tl_if_novalue:nTF 452
\tl_map_break: 617
\tl_map_inline:Nn 612, 614
\tl_new:N 96, 97
\tl_put_right:Nn 621
\tl_set:Nn 585
\tl_set_eq:NN 624
\token 64, 71, 79
tt (key) 3, 3, 98

U

\use:c 112, 146, 284, 288
\usefontset 12, 466
\UTFencname 628

V

variables:
\c__lfc_scope_default_str 87, 224
\g__lfc_lum_done_bool 87, 187, 193, 442, 448
\l__lfc_do_bool 87, 106, 182, 199, 425, 428
\l__lfc_features_seq 87, 217, 223, 226, 230, 307, 316, 322, 396
\l__lfc_scope_seq 87, 325, 327, 374, 381, 386, 391
\l__lfc_scope_str .. 87, 177, 218, 224, 229, 234, 304, 310, 313, 319, 335, 339, 341, 343, 345, 347, 356, 360, 364, 366, 399
\l__lfc_tmpa_bool 87, 613, 616, 620
\l__lfc_tmpa_str 87, 242, 245, 329, 334, 337, 343, 345, 349, 354, 358, 362, 364, 366
\l__lfc_tmpa_tl 87, 101, 103, 104, 168, 170, 171, 175, 179, 180, 196, 197, 198, 611, 621, 624
\l__lfc_tmpb_tl 87, 102, 104, 105, 176, 180, 181, 585, 614
\version@elt 608
\version@list 609

W

weight (key) *8*, *9*
width (key) *8*, *9*