

The LINGUISṬIX bundle

निरंजन

June 19, 2025 (v0.5a)

🏠 <https://ctan.org/pkg/linguistix>

🔗 <https://puszcza.gnu.org.ua/projects/linguistix>

🔗 <https://matrix.to/#/#linguistix:matrix.org>

Abstract

There are quite a few L^AT_EX packages that support typesetting in linguistics, but most of them lack a modern L^AT_EX-like users syntax as well as a programming interface. The LINGUISṬIX bundle fills this gap. It contains several packages enhancing the general support for linguistics in L^AT_EX. This is a comprehensive documentation of the same comprising of three parts. The first one is the general users manual, the second one documents the programming interface of the bundle, whereas the last one is the documented implementation of all the packages.

Contents

1	Introduction	3	7	LINGUISṬIX-FONTS	5
2	Planned	4		Interface... 13; Implementation... 21	
3	Funding	4	8	LINGUISṬIX-IPA	7
4	Acknowledgements	4		Interface... 13; Implementation... 44	
5	LINGUISṬIX-BASE	5	9	LINGUISṬIX-LOGOS	9
	Interface... 12; Implementation... 18			Interface... 15; Implementation... 69	
6	LINGUISṬIX-FIXPEX	5	10	LINGUISṬIX-NFSS	9
	Interface... 13; Implementation... 19			Interface... 15; Implementation... 71	
				GNU Free Documentation License	83

The LINGUISṬIX bundle

Copyright © 2022, 2023, 2024, 2025 निरंजन (hi.niranjan@pm.me)

Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.3 or any later version published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts. A copy of the license is included in the section entitled ‘GNU Free Documentation License’.

Dedicated to Renuka who taught me rigour under the guise of linguistics...

I Introduction

Linguistics is a discipline that studies the phenomenon of language and for this linguists analyse data from languages across the globe. In order to be able to present the data that is collected for this, linguists use several representational methods that lead to a fiasco when their typesetting is considered. In order to understand the complexity of the task at hand, first, let's have a look at some of the problem cases first. If you are an impatient user and are just willing to read the users manual, you may skip reading the current section and start with section 5 and the ones following it.

I.1 Phonetic symbols

Speech sounds are the building blocks of many human languages and the data collected from languages demands an unambiguous method of representation which is served by the International Phonetic Alphabet. For the longest time, the TIPA package (<https://ctan.org/pkg/tipa>) was the one that produced phonetic symbols in \LaTeX . Visually, it matches the default Computer Modern design of \LaTeX , but TIPA is not Unicode. It is set in a legacy encoding. With the recent developments, the New Computer Modern family supports all the IPA characters (even the ones that are missing in TIPA). They are created keeping in mind the principles of Knuth's Computer Modern. Additionally, the family also supports sans serif (recommended in presentations) and mono (recommended in coding context) families. It supports two weights, i.e., book and regular respectively. The book weight is slightly thicker than the regular weight, but the regular one matches the thickness of the Computer Modern design. Because of the increased thickness, the former looks better. The current document, for example, is typeset in the book weight of New Computer Modern. If you are like me, you probably don't like using non- \LaTeX -fonts. The good news is that the requirements of linguistics are very well fulfilled by the recent developments in the New Computer modern family and it *does* belong to the fraternity of \LaTeX -fonts.

Apart from this, there are some other advantages of the New Computer Modern fonts. The IPA distinguishes between [a] and [a], but unfortunately, in Italic shape, the latter is a variant of the former. E.g., `[a\textit{a}]` produces '[aa]'. Whenever an author uses Italic shape for their transcription and use `a`, a wrong IPA symbol is printed with most fonts. This problem was kindly acknowledged by Antonis Tzolomitis, the developer of New Computer Modern. In the stylistic set dedicated for linguistics, the correct shape was added to the Italic shape by him. Thus, `\ipatext{a\textit{a}}` (a command from `LINGUIS\X-ipa`) renders '[aa]'. The package enables New Computer Modern family with stylistic set `o5` dedicated for IPA. It also adds the brackets or slashes around the argument as explained in section 8.

A similar problem is with the character `g`. E.g., `[g\textit{g}]` produces '[gg]'. Here, the situation is the other way round. The upright 'g' is not recognised by the IPA. The IPA charts generally have the upright version of the Italic shape. To see what this means, try `\ipatext{g\textit{g}}`. It produces [gg] and not [gg].

In order to avail all of these features, I have set New Computer Modern as the default font-family of `LINGUIS\X`. The bundle provides options to control these defaults. Users can use their preferred text and IPA fonts. There also is an option to use the regular weight of NewCM instead of the book weight.

2 Planned

I plan to develop this bundle further in order to support the typesetting of good quality examples with interlinear glossing. My model is to imitate the output of the `expex` package, but with a modern \LaTeX -like syntax. I also plan to provide support for glossing. Currently the `leipzig` package is used, but it has some unresolved bugs. Some syntactic improvements are also possible, I believe.

3 Funding

I am a doctorate student without a fellowship (thanks to our education policies!) currently sustaining only with a full time job unrelated to linguistics that consumes most of my working hours. At times, it becomes difficult to continue the research, the job and the passion development projects. `LINGUIS \TeX` needs funding in order to sustain. If you think you can support it, you can contact me on the email ID found on the front page.

As of 2025-05-29, I have recieved funding from the \TeX users group's \TeX development fund. They have decided to support the development of 'linguistix-glossing' (the logo will be available once the package is ready).

4 Acknowledgements

This package relies the most on the New Computer Modern font family. I would like to express my gratitude to Antonis Tsolomitis who tirelessly worked on the set of IPA symbols and brought back the good old charm of TIPA's design in the modern Unicode world. I would like to thank Renuka and Avinash who taught me linguistics. They nourished my passion, helped me pursue my love for the subject as well as the computation that came along with it. I could have never imagined myself working on a package written in \LaTeX 's syntax. Not so long ago, I used to find it very complicated. It's mostly Jonathan Spratte and Florent Rougon's help (and, at times, scolding :P) that helped me get used to it. I would also like to mention C.V. Radhakrishnan for being an important part of my journey in \LaTeX . Lastly, to all the free software people who have created this friendly and supportive world for people by investing their precious time and resources!

Hardly in a week after the initial release, the \TeX users group decided to financially support the development of a planned package in the bundle. I am grateful to them for their support.

Documentation

The bundle is comprised of several packages that are developed for different purposes. In order to load all the packages of the bundle, one can issue:

```
\usepackage{linguistix}
```

This is the easiest method for getting all of `LINGUIS \TeX` in one go. But, if you don't need all the packages of the bundle, you may load the required packages separately. We will start with the elementary package that sets up things for other packages of the bundle.

5 LINGUIS \mathcal{T} \mathbf{I} \mathbf{X} -BASE

L^AT_EX₃-interface | Implementation

This package provides a single command that is used in all the other packages of the bundle. The command is:

`\linguistix` $\{(key-value-list)\}$

We have a single set of keys for the entire bundle. Each package appends keys to the same set. The argument of this central processor command is the comma-separated $\langle key-value-list \rangle$. So you can load any package of LINGUIS \mathcal{T} \mathbf{I} \mathbf{X} and use the `\linguistix` command. The only exception to this is LINGUIS \mathcal{T} \mathbf{I} \mathbf{X} -NFSS. We will see how it is different in its section.

6 LINGUIS \mathcal{T} \mathbf{I} \mathbf{X} -FIXPEX

L^AT_EX₃-interface | Implementation

This package offers a fix for the clash between `expex` and `unicode-math`. It provides a single command.

`\umgla` This is a replica of the `unicode-math-\gla`. Since the `expex-\gla` is more relevant in linguistics, I set it as the default. If one needs to use `unicode-math-\gla`, they can use this command.

7 LINGUIS \mathcal{T} \mathbf{I} \mathbf{X} -FONTS

L^AT_EX₃-interface | Implementation

This is a package that loads the New Computer Modern family for the entire document. The package sets fonts for both text and math. It has keys for customisation for both. Note that just loading this package does *not* provide any support for IPA. For that one needs LINGUIS \mathcal{T} \mathbf{I} \mathbf{X} -IPA separately.

Antonis suggested a typographic enhancement for the logo of L^AT_EX. The default logo scales the ‘A’ and that affects the ‘colour’ of the font. This is why I renew the logo with the code given by Antonis. The original logo is also available with an alternative command.

`\LaTeX` L^AT_EX
`\ogLaTeX` L^AT_EX

The package provides only these commands. Let’s now have a look at the keys provided for the text.

7.1 Text

Most keys of this package are prefixed with the `text` in order to distinguish them from the maths and IPA ones. There aren’t any commands provided by the package. Most of the important features of the `fontspec` package are variablised with `l3keys`.

The ‘old style numbers’ have varying heights. Some numbers have ascenders and some have descenders (e.g., 6789). According to Bringhurst, 2004, this makes them easier to read in running text. Lining numbers, on the other hand have uniform heights. They go well with all capital text (rare). Thus, for the general text, I enable this setting by default in LINGUIS \mathcal{T} \mathbf{I} \mathbf{X} -FONTS.

Apart from that, the New Computer Modern font family provides an old-style shape for the number ‘1’ (this exact shape!), but it is provided as a character variant. Different fonts may use these arbitrary slots for any character’s alternation. Therefore this setting should not be loaded blindly. Let’s have a look at the keys that can be employed to change these behaviours.

<code>old style numbers</code>	<code>= {\langle truth value \rangle}</code>	<code>true false</code>
<code>old style one</code>	<code>= {\langle truth value \rangle}</code>	<code>true false</code>

If one wants to disable old style numbers, they may use the `old style numbers` key with the `false` value (default is `true`)¹. Note that printing of old style numbers also depends on whether the font you select has old style numbers or not. The relevant settings are added by the package to the font automatically, but while selecting the font, make sure whether the old style table is present in the font or not.

Suppose one wants the alternative shape of number ‘1’ from the New Computer Modern family, they may use the key `old style one` (default is `false`; adding `true` is optional).

Let’s have a look at the three way distinction we get because of this.

0123456789	Old style with default 1
0I23456789	Old style with the old 1
0123456789	Lining

<code>newcm</code>
<code>newcm sans</code>
<code>newcm mono</code>
<code>newcm regular</code>
<code>newcm regular sans</code>
<code>newcm regular mono</code>

These are some keys that come in handy for setting New Computer Modern defaults. All the necessary values are stored in these. The keys that have **regular** in their names refer to the ‘regular’ variants of New Computer Modern fonts. These variants match the colour and widths of the Latin Modern fonts. One may use these keys to override the changed defaults.

7.2 Maths

LINGUISCIX-FONTS sets maths fonts also. In order to control the settings related to maths, the following keys can be used.

<code>math</code>	<code>= {\langle math font \rangle}</code>
<code>math features</code>	<code>= {\langle math font features \rangle}</code>
<code>math bold</code>	<code>= {\langle bold math font \rangle}</code>
<code>math bold features</code>	<code>= {\langle bold math font features \rangle}</code>

The `math` and `math bold` keys set the respective fonts (i.e., regular and bold fonts for mathematics respectively). The keys suffixed with `features` set the font features of the same.

¹The possible and the default values of keys are given at the right side in the documentation and the defaults are highlighted in red.

`bourbaki's empty set` = `{\truth value}` `true` | `false`

In (L^A)T_EX, the default shape of the ‘empty set’ symbol is: ‘ \emptyset ’, but the symbol used by the Bourbaki group is still considered more correct and preferred by many (including me). New Computer Modern Math fonts provide it as a character variant that I activate by default. Thus `\emptyset` always renders: ‘ \emptyset ’ and not: ‘ \empty ’. In order to change this behaviour, one may use this key and set it to false for getting the slashed-zero of original (L^A)T_EX. Hail plumbers union, *IYKYK!* ;-)

8 LINGUIS τ IX-ipa

L^AT_EX3-interface | Implementation

This package sets the fonts exclusively for the IPA. The commands provided for switching to the IPA control all serif, sans serif and typewriter families. This package can be loaded standalone for loading IPA fonts as well as some switch commands useful in running text. New Computer Modern provides a special stylistic set dedicated for linguistics. It is enabled for IPA fonts automatically with this package. Only the legally marked up IPA is affected by the customisation provided by this package. For switching to the IPA, LINGUIS τ IX-ipa provides one command with a starred variant.

`\ipatext` `{\phonetic transcription}`
`\ipatext*` `{\phonemic transcription}`

This is a command that resembles with the TIPA command `\textipa`. I have deliberately kept it distinct from it so that just in case somebody wants to use their old TIPA code in a Unicode document, the commands won’t clash (I highly discourage doing this, though). The command comes with a starred variant. The behaviour of the unstarred command is to print the argument in brackets for phonetic transcription, e.g.: `\ipatext{a1 phi: e1}` \rightarrow $[a_1 p^{hi}: e_1]$ whereas the starred version prints it in slashes for phonemic transcription, e.g.: `\ipatext*{a1 phi: e1}` \rightarrow $/a_1 p^{hi}: e_1/$.

Suppose someone just wants to load the font without the brackets or slashes, they can use the following command for switching to the IPA without adding the aforementioned.

`\lngxipa` This also is a command that switches to the IPA-only features (default as well as user added). This command, of course, leaks and that’s why *should* be delimited. E.g., the following code lines produce $[a_1 p^{hi}: e_1]$ and $/a_1 p^{hi}: e_1/$ respectively:

`{\lngxipa [a1 phi: e1]}`
`{\lngxipa /a1 phi: e1/}`

`ipa newcm`
`ipa newcm sans`
`ipa newcm mono`
`ipa newcm regular`
`ipa newcm regular sans`
`ipa newcm regular mono`

All the IPA fonts are stored in variables as seen in table 1 and table 2. These keys reset the IPA-only fonts to New Computer Modern. They can be used even for resetting to New Computer Modern from another IPA font. In order to change or reset to the IPA defaults these keys can be used. They store the names of the New Computer Modern font family in the variables concerning IPA. The keys that contain **regular** in their name use the regular version of New Computer Modern that matches the colour of Latin Modern.

Let’s now see the combined table of font keys provided by both LINGUIS τ IX-FONTS and LINGUIS τ IX-ipa.

Family	LINGUIS _{CL} X-FONTS	LINGUIS _{CL} X-IPA
Serif	text upright	ipa upright
	text upright features	ipa upright features
	text bold upright	ipa bold upright
	text bold upright features	ipa bold upright features
	text italic	ipa italic
	text italic features	ipa italic features
	text bold italic	ipa bold italic
	text bold italic features	ipa bold italic features
	text slanted	ipa slanted
	text slanted features	ipa slanted features
	text bold slanted	ipa bold slanted
	text bold slanted features	ipa bold slanted features
	text swash	ipa swash
	text swash features	ipa swash features
	text bold swash	ipa bold swash
	text bold swash features	ipa bold swash features
	text small caps	ipa small caps
	text small caps features	ipa small caps features
Sans serif	text sans upright	ipa sans upright
	text sans upright features	ipa sans upright features
	text sans bold upright	ipa sans bold upright
	text sans bold upright features	ipa sans bold upright features
	text sans italic	ipa sans italic
	text sans italic features	ipa sans italic features
	text sans bold italic	ipa sans bold italic
	text sans bold italic features	ipa sans bold italic features
	text sans slanted	ipa sans slanted
	text sans slanted features	ipa sans slanted features
	text sans bold slanted	ipa sans bold slanted
	text sans bold slanted features	ipa sans bold slanted features
	text sans swash	ipa sans swash
	text sans swash features	ipa sans swash features
	text sans bold swash	ipa sans bold swash
	text sans bold swash features	ipa sans bold swash features
	text sans small caps	ipa sans small caps
	text sans small caps features	ipa sans small caps features
Monospaced	text mono upright	ipa mono upright
	text mono upright features	ipa mono upright features
	text mono bold upright	ipa mono bold upright
	text mono bold upright features	ipa mono bold upright features
	text mono italic	ipa mono italic
	text mono italic features	ipa mono italic features
	text mono bold italic	ipa mono bold italic
	text mono bold italic features	ipa mono bold italic features
	text mono slanted	ipa mono slanted

Continued on the next page...

Family	LINGUISŢI $\textcolor{violet}{X}$ -FONTS	LINGUISŢI $\textcolor{violet}{X}$ -IPA
	text mono slanted features	ipa mono slanted features
	text mono bold slanted	ipa mono bold slanted
	text mono bold slanted features	ipa mono bold slanted features
	text mono swash	ipa mono swash
	text mono swash features	ipa mono swash features
	text mono bold swash	ipa mono bold swash
	text mono bold swash features	ipa mono bold swash features
	text mono small caps	ipa mono small caps
	text mono small caps features	ipa mono small caps features
<i>End of the table...</i>		

Table 1: Font keys provided by LINGUISŢI $\textcolor{violet}{X}$ -FONTS and LINGUISŢI $\textcolor{violet}{X}$ -IPA

9 LINGUISŢI $\textcolor{violet}{X}$ -LOGOS

L^AT_EX₃-interface | Implementation

This is a small package that provides commands for printing logos of the LINGUISŢI $\textcolor{violet}{X}$ bundle. The logo is printed in New Computer Modern Uncial font. It uses purple colour for the ‘X’ in it and it is defined using `l3color` module. It provides one command that takes an optional argument. Obviously it is ‘protected’. It is as follows:

`\lngxlogo` [*package name*]

The logo of the *package name* from the LINGUISŢI $\textcolor{violet}{X}$ bundle is printed with this command, e.g., `\lngxlogo{fonts}` \rightarrow LINGUISŢI $\textcolor{violet}{X}$ fonts.

Sometimes, the logos might be required to be used in an expandable way, but optional arguments are not supported in expandable commands. Thus we create separate commands for separate packages. Even these ones have the `lngx` prefix. It is followed by the package name, e.g., `fonts` or `ipa` and finally the suffix `logo`. In the context of `hyperref`, their behaviour is different than in the context of normal text. The commands and their are as follows:

<code>\lngxpkg</code>	★	LINGUISŢI $\textcolor{violet}{X}$
<code>\lngxbaselogo</code>	★	LINGUISŢI $\textcolor{violet}{X}$ -BASE
<code>\lngxfontlogo</code>	★	LINGUISŢI $\textcolor{violet}{X}$ -FONTS
<code>\lngxipalogo</code>	★	LINGUISŢI $\textcolor{violet}{X}$ -IPA
<code>\lngxlogoslogo</code>	★	LINGUISŢI $\textcolor{violet}{X}$ -LOGOS
<code>\lngxnfsslogo</code>	★	LINGUISŢI $\textcolor{violet}{X}$ -NFSS

10 LINGUISŢI $\textcolor{violet}{X}$ -NFSS

L^AT_EX₃-interface | Implementation

This is an extension package to the existing NFSS scheme of L^AT_EX. The NFSS mainly works on the four facets of the text.

1. Encoding
2. Family

3. Shape

4. Series

These facets are reset to default by the `\normalfont` and `\selectfont` commands. These commands work on some internals that are reset with every usage of some commands that set them, e.g., `\rmfamily`, `\bfseries`. There isn't any way to control this unless some internals are touched and there might be incidences where one does want to control them, e.g., try compiling the following code in Lua^AT_EX.

```
\documentclass{article}

\begin{document}
\makeatletter
\fontencoding{OT1}\sffamily\itshape\bfseries
\selectfont
\fontencoding\ | \fontfamily\ | \fontseries\ | \fontshape\quad
\normalfont
\fontencoding\ | \fontfamily\ | \fontseries\ | \fontshape
\end{document}
```

As can be seen in the output, the first line shows the text in OT1 encoding, sans family, bold series and Italic shape. After `\normalfont`, every aspect of the text is reset to the default one. The default encoding is TU. We can see TU instead of OT1 after `\normalfont`. So is the case with family (default: `\rmfamily`), series (default: `\mdseries`) and shape (default: `\upshape`). This usually is okay, but sometimes it doesn't fit the requirement. E.g., the following might be used with the intention of switching from the IPA font to the text font, but as can be seen, it doesn't really change anything.

```
\documentclass{article}
\usepackage{linguistix-fonts}
\usepackage{linguistix-ipa}
\linguistix{%
  text upright          = {KpRoman-Regular.otf},%
  text upright features = {Color={green}},%
  ipa upright           = {KpSans-Regular.otf},%
  ipa upright features  = {Color={red}}}%
}

\begin{document}
test \lngxipa test \normalfont test
\end{document}
```

The reason for this is the way `\lngxipa` is defined. It resets `\rmdefault`, `\sfdefault` and `\ttdefault` and uses `\normalfont` to initialise this new super font family (see: <https://tex.stackexchange.com/a/729805>). Setting a 'super' font family effectively

changes the behaviour of `\normalfont` permanently. By the way, this is not just something that `LINGUISCIX` has to deal with. This situation may arise whenever one wants to have a font family command that sets all serif, sans serif and monospaced font families. `LINGUISCIX-NFSS` is useful in such cases. It introduces the concept of ‘super’ font family. It shouldn’t be confused with \LaTeX 2_ϵ ’s ‘meta’ font family. It refers to `rm`, `sf` or `tt` in the kernel. Note that, as of now, \LaTeX 2_ϵ does *not* provide any public interface to save ‘meta’ family, as well as, the current encoding, series and shape. This package provides control over these facets. Let’s have a look at the macros it provides.

<hr/>	
<code>\IfEncodingTF</code>	$\star \{ \langle encoding \rangle \} \{ \langle true code \rangle \} \{ \langle false code \rangle \}$
<code>\IfEncodingT</code>	$\star \{ \langle encoding \rangle \} \{ \langle true code \rangle \}$
<code>\IfEncodingF</code>	$\star \{ \langle encoding \rangle \} \{ \langle false code \rangle \}$
<code>\CurrentEncoding</code>	\star If the current encoding matches with the given $\langle encoding \rangle$, it selects the true branch; false otherwise. The <code>\CurrentEncoding</code> macro expands to the current encoding.
<hr/>	
<code>\IfMetaFamilyTF</code>	$\star \{ \langle meta family \rangle \} \{ \langle true code \rangle \} \{ \langle false code \rangle \}$
<code>\IfMetaFamilyT</code>	$\star \{ \langle meta family \rangle \} \{ \langle true code \rangle \}$
<code>\IfMetaFamilyF</code>	$\star \{ \langle meta family \rangle \} \{ \langle false code \rangle \}$
<code>\CurrentMetaFamily</code>	\star If the current meta family matches with the given $\langle meta family \rangle$, it selects the true branch; false otherwise. The <code>\CurrentMetaFamily</code> macro expands to the current meta family.
<hr/>	
<code>\IfSuperFamilyTF</code>	$\star \{ \langle super family \rangle \} \{ \langle true code \rangle \} \{ \langle false code \rangle \}$
<code>\IfSuperFamilyT</code>	$\star \{ \langle super family \rangle \} \{ \langle true code \rangle \}$
<code>\IfSuperFamilyF</code>	$\star \{ \langle super family \rangle \} \{ \langle false code \rangle \}$
<code>\CurrentSuperFamily</code>	\star If the current super family matches with the given $\langle super family \rangle$, it selects the true branch; false otherwise. The <code>\CurrentSuperFamily</code> macro expands to the current super family.
<hr/>	
<code>\IfSeriesTF</code>	$\star \{ \langle series \rangle \} \{ \langle true code \rangle \} \{ \langle false code \rangle \}$
<code>\IfSeriesT</code>	$\star \{ \langle series \rangle \} \{ \langle true code \rangle \}$
<code>\IfSeriesF</code>	$\star \{ \langle series \rangle \} \{ \langle false code \rangle \}$
<code>\CurrentSeries</code>	\star If the current series matches with the given $\langle series \rangle$, it selects the true branch and false otherwise. The <code>\CurrentSeries</code> macro expands to the current series.
<hr/>	
<code>\IfShapeTF</code>	$\star \{ \langle shape \rangle \} \{ \langle true code \rangle \} \{ \langle false code \rangle \}$
<code>\IfShapeT</code>	$\star \{ \langle shape \rangle \} \{ \langle true code \rangle \}$
<code>\IfShapeF</code>	$\star \{ \langle shape \rangle \} \{ \langle false code \rangle \}$
<code>\CurrentShape</code>	\star If the current series matches with the given $\langle shape \rangle$, it selects the true branch and false otherwise. The <code>\CurrentShape</code> macro expands to the current shape.
<hr/>	
<code>\superfontfamily</code>	$\{ \langle family id \rangle \} \{ \langle rm=\{ \langle rm nfss \rangle \}, sf=\{ \langle sf nfss \rangle \}, tt=\{ \langle tt nfss \rangle \} \}$

Every super font family has a $\langle family id \rangle$, even the default one (i.e., `default`). This command creates a super family with the given $\langle family id \rangle$ s. The $\langle meta family keys \rangle$ argument accepts a list of specific keys, `rm`, `sf` and `tt`. They take the NFSS family names of these meta families as arguments. One may define a font with, say, `\newfontfamily`, pass the `NFSSkeys=\{ \langle key \rangle \}` option to it and use the $\langle key \rangle$ in the suitable $\langle meta family key \rangle$. Note that using all these keys is *not* mandatory. A super family may have ≤ 3 keys.

<code>\softsuperfontfamily</code>	<code>{\langle id \rangle}{\langle encoding, family, series, shape \rangle}</code>
<code>\softersuperfontfamily</code>	<code>{\langle id \rangle}</code>
<code>\softtestsuperfontfamily</code>	<code>{\langle id \rangle}</code>

These commands loads the super font family with the given $\langle id \rangle$. The attributes listed in the second argument are the only choices available. The required super font family is loaded and the listed attributes are reset to the ones that were active before. All the four are not required. The number of attributes may be ≤ 4 . The `\softernormalfont` command excludes encoding and reactivates all the other attributes, whereas the `\softestnormalfont` command reactivates all of them.

<code>\softnormalfont</code>	<code>{\langle encoding, family, series, shape \rangle}</code>
<code>\softernormalfont</code>	
<code>\softestnormalfont</code>	

Similar to `\softsuperfontfamily` and friends, these commands switch back to the default super font family, but reactivate the previously active font attributes. The argument to `\softnormalfont` takes the list of the required font attributes. It can have ≤ 4 values. Now try the following example:

```

\documentclass{article}
\usepackage{linguistix}
\linguistix{%
  text upright features = {Color={green}},%
  ipa upright features  = {Color={red}}}%
}

\begin{document}
test \lngxipa test \softernormalfont test\par
\makeatletter
\sffamily\itshape\bfseries
\f@family\ | \f@series\ | \f@shape\quad
\softnormalfont{series}
\f@family\ | \f@series\ | \f@shape
\end{document}

```

Better? :-)

L^AT_EX₃ interface for programmers

In this section, we take a look at the public L^AT_EX₃ commands of the bundle. These can be considered stable and can be used in production code.

LINGUISTIX-BASE

[Documentation](#) | [Implementation](#)

<code>\lngx_set_keys:n</code>	$\langle keyval list \rangle$
-------------------------------	-------------------------------

This is the base command for `\linguistix`. It takes a comma separated list of $\langle keyval list \rangle$ and parses it.

LINGUIS \mathcal{T} X-FIXPEX

[Documentation](#) | [Implementation](#)

No L^AT_EX3 function provided by this package.

LINGUIS \mathcal{T} X-FONTS

[Documentation](#) | [Implementation](#)

<hr/> <code>\g_lngx_old_style_bool</code> <code>\g_lngx_old_style_one_bool</code> <code>\g_lngx_bourbaki_bool</code> <hr/>	These are the two booleans that are used to check if the old style numbers, the old style one (i.e., ‘r’) and Bourbaki’s empty set symbol (i.e., ‘ \emptyset ’) is asked by the user.
----------------------------------------------------------------------------------------------------------------------------------	-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

<hr/> <code>\lngx_set_main_font:nn</code> $\{\langle features \rangle\}$ $\{\langle font \rangle\}$ <code>\lngx_set_main_font:ee</code> $\{\langle features \rangle\}$ $\{\langle font \rangle\}$ <code>\lngx_set_sans_font:nn</code> $\{\langle features \rangle\}$ $\{\langle font \rangle\}$ <code>\lngx_set_sans_font:ee</code> $\{\langle features \rangle\}$ $\{\langle font \rangle\}$ <code>\lngx_set_mono_font:nn</code> <code>\lngx_set_mono_font:ee</code> <code>\lngx_set_math_font:nn</code> <code>\lngx_set_math_font:ee</code> <hr/>	These commands take two arguments, expand them if the :ee variant is used. These are wrapper commands around the font-setting commands of fontspec and unicode-math, i.e., <code>\setmainfont</code> , <code>\setsansfont</code> , <code>\setmonofont</code> and <code>\setmathfont</code> . The $\langle features \rangle$ are passed to the optional argument and the $\langle font \rangle$ is passed to the mandatory argument of the respective command from the aforementioned list.
--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------	--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

LINGUIS \mathcal{T} X-IPA

[Documentation](#) | [Implementation](#)

This package provides a few wrapper functions around fontspec’s commands.

<hr/> <code>\lngx_set_main_ipa_font:nn</code> $\{\langle features \rangle\}$ $\{\langle font \rangle\}$ <code>\lngx_set_main_ipa_font:ee</code> <code>\lngx_main_ipa:</code> <code>lngx_ipa_rm_nfss</code> <hr/>	These functions set the IPA fonts for the serif variants. The $\langle font \rangle$ is set with $\langle features \rangle$ for the serif IPA. The command to switch to this family is <code>\lngx_main_ipa:</code> . It can be accessed with the NFSS family <code>lngx_ipa_rm_nfss</code> .
---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------	-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

<hr/> <code>\lngx_set_sans_ipa_font:nn</code> $\{\langle features \rangle\}$ $\{\langle font \rangle\}$ <code>\lngx_set_sans_ipa_font:ee</code> <code>\lngx_sans_ipa:</code> <code>lngx_ipa_sf_nfss</code> <hr/>	These functions set the IPA fonts for the sans variants. The $\langle font \rangle$ is set with $\langle features \rangle$ for the sans IPA. The command to switch to this family is <code>\lngx_sans_ipa:</code> . It can be accessed with the NFSS family <code>lngx_ipa_sf_nfss</code> .
---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------	---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

<hr/> <code>\lngx_set_mono_ipa_font:nn</code> $\{\langle features \rangle\}$ $\{\langle font \rangle\}$ <code>\lngx_set_mono_ipa_font:ee</code> <code>\lngx_mono_ipa:</code> <code>lngx_ipa_tt_nfss</code> <hr/>	These functions set the IPA fonts for the mono variants. The $\langle font \rangle$ is set with $\langle features \rangle$ for the mono IPA. The command to switch to this family is <code>\lngx_mono_ipa:</code> . It can be accessed with the NFSS family <code>lngx_ipa_nfss_nfss</code> .
---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------	-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

<hr/> <code>\lngx_ipa:</code> <code>lngx_ipa</code> <hr/>	The <code>\lngx_ipa:</code> command loads the super family <code>lngx_ipa</code> (see the documentation of LINGUIS \mathcal{T} X-NFSS). The <code>\lngx_ipa:</code> function has a user-side command <code>\lngxipa</code> too.
--------------------------------------------------------------	-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

Variables for fonts and features

Now we look at the table that summarises the `tl`s that are used by the package for saving serif, sans serif and monospaced fonts and their features. Note that this table also lists the `tl`s used by the `LINGUISTLX-ipa` package.



Serif	Sans serif	Monospaced
<code>\g_lngx_text_upright_tl</code>	<code>\g_lngx_text_sans_upright_tl</code>	<code>\g_lngx_text_mono_upright_tl</code>
<code>\g_lngx_ipa_upright_tl</code>	<code>\g_lngx_ipa_sans_upright_tl</code>	<code>\g_lngx_ipa_mono_upright_tl</code>
<code>\g_lngx_text_upright_features_tl</code>	<code>\g_lngx_text_sans_upright_features_tl</code>	<code>\g_lngx_text_mono_upright_features_tl</code>
<code>\g_lngx_ipa_upright_features_tl</code>	<code>\g_lngx_ipa_sans_upright_features_tl</code>	<code>\g_lngx_ipa_mono_upright_features_tl</code>
<code>\g_lngx_text_bold_upright_tl</code>	<code>\g_lngx_text_sans_bold_upright_tl</code>	<code>\g_lngx_text_mono_bold_upright_tl</code>
<code>\g_lngx_ipa_bold_upright_tl</code>	<code>\g_lngx_ipa_sans_bold_upright_tl</code>	<code>\g_lngx_ipa_mono_bold_upright_tl</code>
<code>\g_lngx_text_bold_upright_features_tl</code>	<code>\g_lngx_text_sans_bold_upright_features_tl</code>	<code>\g_lngx_text_mono_bold_upright_features_tl</code>
<code>\g_lngx_ipa_bold_upright_features_tl</code>	<code>\g_lngx_ipa_sans_bold_upright_features_tl</code>	<code>\g_lngx_ipa_mono_bold_upright_features_tl</code>
<code>\g_lngx_text_italic_tl</code>	<code>\g_lngx_text_sans_italic_tl</code>	<code>\g_lngx_text_mono_italic_tl</code>
<code>\g_lngx_ipa_italic_tl</code>	<code>\g_lngx_ipa_sans_italic_tl</code>	<code>\g_lngx_ipa_mono_italic_tl</code>
<code>\g_lngx_text_italic_features_tl</code>	<code>\g_lngx_text_sans_italic_features_tl</code>	<code>\g_lngx_text_mono_italic_features_tl</code>
<code>\g_lngx_ipa_italic_features_tl</code>	<code>\g_lngx_ipa_sans_italic_features_tl</code>	<code>\g_lngx_ipa_mono_italic_features_tl</code>
<code>\g_lngx_text_bold_italic_tl</code>	<code>\g_lngx_text_sans_bold_italic_tl</code>	<code>\g_lngx_text_mono_bold_italic_tl</code>
<code>\g_lngx_ipa_bold_italic_tl</code>	<code>\g_lngx_ipa_sans_bold_italic_tl</code>	<code>\g_lngx_ipa_mono_bold_italic_tl</code>
<code>\g_lngx_text_bold_italic_features_tl</code>	<code>\g_lngx_text_sans_bold_italic_features_tl</code>	<code>\g_lngx_text_mono_bold_italic_features_tl</code>
<code>\g_lngx_ipa_bold_italic_features_tl</code>	<code>\g_lngx_ipa_sans_bold_italic_features_tl</code>	<code>\g_lngx_ipa_mono_bold_italic_features_tl</code>
<code>\g_lngx_text_slanted_tl</code>	<code>\g_lngx_text_sans_slanted_tl</code>	<code>\g_lngx_text_mono_slanted_tl</code>
<code>\g_lngx_ipa_slanted_tl</code>	<code>\g_lngx_ipa_sans_slanted_tl</code>	<code>\g_lngx_ipa_mono_slanted_tl</code>
<code>\g_lngx_text_slanted_features_tl</code>	<code>\g_lngx_text_sans_slanted_features_tl</code>	<code>\g_lngx_text_mono_slanted_features_tl</code>
<code>\g_lngx_ipa_slanted_features_tl</code>	<code>\g_lngx_ipa_sans_slanted_features_tl</code>	<code>\g_lngx_ipa_mono_slanted_features_tl</code>
<code>\g_lngx_text_bold_slanted_tl</code>	<code>\g_lngx_text_sans_bold_slanted_tl</code>	<code>\g_lngx_text_mono_bold_slanted_tl</code>
<code>\g_lngx_ipa_bold_slanted_tl</code>	<code>\g_lngx_ipa_sans_bold_slanted_tl</code>	<code>\g_lngx_ipa_mono_bold_slanted_tl</code>
<code>\g_lngx_text_bold_slanted_features_tl</code>	<code>\g_lngx_text_sans_bold_slanted_features_tl</code>	<code>\g_lngx_text_mono_bold_slanted_features_tl</code>
<code>\g_lngx_ipa_bold_slanted_features_tl</code>	<code>\g_lngx_ipa_sans_bold_slanted_features_tl</code>	<code>\g_lngx_ipa_mono_bold_slanted_features_tl</code>
<code>\g_lngx_text_swash_tl</code>	<code>\g_lngx_text_sans_swash_tl</code>	<code>\g_lngx_text_mono_swash_tl</code>
<code>\g_lngx_ipa_swash_tl</code>	<code>\g_lngx_ipa_sans_swash_tl</code>	<code>\g_lngx_ipa_mono_swash_tl</code>
<code>\g_lngx_text_swash_features_tl</code>	<code>\g_lngx_text_sans_swash_features_tl</code>	<code>\g_lngx_text_mono_swash_features_tl</code>
<code>\g_lngx_ipa_swash_features_tl</code>	<code>\g_lngx_ipa_sans_swash_features_tl</code>	<code>\g_lngx_ipa_mono_swash_features_tl</code>
<code>\g_lngx_text_bold_swash_tl</code>	<code>\g_lngx_text_sans_bold_swash_tl</code>	<code>\g_lngx_text_mono_bold_swash_tl</code>
<code>\g_lngx_ipa_bold_swash_tl</code>	<code>\g_lngx_ipa_sans_bold_swash_tl</code>	<code>\g_lngx_ipa_mono_bold_swash_tl</code>
<code>\g_lngx_text_bold_swash_features_tl</code>	<code>\g_lngx_text_sans_bold_swash_features_tl</code>	<code>\g_lngx_text_mono_bold_swash_features_tl</code>
<code>\g_lngx_ipa_bold_swash_features_tl</code>	<code>\g_lngx_ipa_sans_bold_swash_features_tl</code>	<code>\g_lngx_ipa_mono_bold_swash_features_tl</code>
<code>\g_lngx_text_small_caps_tl</code>	<code>\g_lngx_text_sans_small_caps_tl</code>	<code>\g_lngx_text_mono_small_caps_tl</code>
<code>\g_lngx_ipa_small_caps_tl</code>	<code>\g_lngx_ipa_sans_small_caps_tl</code>	<code>\g_lngx_ipa_mono_small_caps_tl</code>
<code>\g_lngx_text_small_caps_features_tl</code>	<code>\g_lngx_text_sans_small_caps_features_tl</code>	<code>\g_lngx_text_mono_small_caps_features_tl</code>
<code>\g_lngx_ipa_small_caps_features_tl</code>	<code>\g_lngx_ipa_sans_small_caps_features_tl</code>	<code>\g_lngx_ipa_mono_small_caps_features_tl</code>

End of the table...

Table 2: Variables for fonts and font features provided by `LINGUISTLX-FONTS` and `LINGUISTLX-ipa`

There are only two L^AT_EX₃ functions provided by this package.

`\lngx_logo_font:` This function switches to the New Computer Modern Uncial font family.

`lngx_purple_color` I don't like the default purple colour of the `xcolor` package (i.e., ) . Thus I have created a new colour using `!3color` module. It can be accessed using this variable. The color looks like: .

LINGUIS \mathcal{T} **X**-NFSS

This subsection discusses the programming interface LINGUIS \mathcal{T} **X**-NFSS provides.

`\c_lngx_default_rmdefault_tl` * These `tl`s expand to the default values of the fonts set at the `\begindocument/end`
`\c_lngx_default_sfdefault_tl` * hook. These are not supposed to be changed and hence they are set with the `c` prefix.
`\c_lngx_default_ttdefault_tl` *

`\l_lngx_current_encoding_tl` * These `tl`s expand to the current values of encoding, meta family, super family,
`\l_lngx_current_meta_family_tl` * series and shape respectively. Note that these are updated time to time by the
`\l_lngx_current_super_family_tl` * commands that change them (package-internal or L^AT_EX-internal).
`\l_lngx_current_series_tl` *
`\l_lngx_current_shape_tl` *

`\lngx_if_encoding_p:n` * $\{\langle encoding \rangle\}$
`\lngx_if_encoding:nTF` * $\{\langle encoding \rangle\}\{\langle true code \rangle\}\{\langle false code \rangle\}$
`\lngx_if_meta_family_p:n` * $\{\langle meta font family \rangle\}$
`\lngx_if_meta_family:nTF` * $\{\langle meta font family \rangle\}\{\langle true code \rangle\}\{\langle false code \rangle\}$
`\lngx_if_super_family_p:n` * $\{\langle super font family \rangle\}$
`\lngx_if_super_family:nTF` * $\{\langle super font family \rangle\}\{\langle true code \rangle\}\{\langle false code \rangle\}$
`\lngx_if_series_p:n` * $\{\langle series \rangle\}$
`\lngx_if_series:nTF` * $\{\langle series \rangle\}\{\langle true code \rangle\}\{\langle false code \rangle\}$
`\lngx_if_shape_p:n` * $\{\langle shape \rangle\}$
`\lngx_if_shape:nTF` * $\{\langle shape \rangle\}\{\langle true code \rangle\}\{\langle false code \rangle\}$

`\lngx_if_meta_family_rm_p:` *
`\lngx_if_meta_family_rm:TF` * $\{\langle true code \rangle\}\{\langle false code \rangle\}$
`\lngx_if_meta_family_sf_p:` *
`\lngx_if_meta_family_sf:TF` * $\{\langle true code \rangle\}\{\langle false code \rangle\}$
`\lngx_if_meta_family_tt_p:` *
`\lngx_if_meta_family_tt:TF` * $\{\langle true code \rangle\}\{\langle false code \rangle\}$

These conditionals select the true branch if the `rm`, `sf`, `tt` families (respectively) are active, false otherwise.

```

\lngx_if_series_md_p: *
\lngx_if_series_md:TF * {\langle true code \rangle}{\langle false code \rangle}
\lngx_if_series_bf_p: *
\lngx_if_series_bf:TF * {\langle true code \rangle}{\langle false code \rangle}

```

These conditionals select the true branch if the `md`, `bf` series (respectively) are active, false otherwise.

```

\lngx_if_shape_up_p: *
\lngx_if_shape_up:TF * {\langle true code \rangle}{\langle false code \rangle}
\lngx_if_shape_it_p: *
\lngx_if_shape_it:TF * {\langle true code \rangle}{\langle false code \rangle}
\lngx_if_shape_sc_p: *
\lngx_if_shape_sc:TF * {\langle true code \rangle}{\langle false code \rangle}
\lngx_if_shape_ssc_p: *
\lngx_if_shape_ssc:TF * {\langle true code \rangle}{\langle false code \rangle}
\lngx_if_shape_sl_p: *
\lngx_if_shape_sl:TF * {\langle true code \rangle}{\langle false code \rangle}
\lngx_if_shape_sw_p: *
\lngx_if_shape_sw:TF * {\langle true code \rangle}{\langle false code \rangle}
\lngx_if_shape_ulc_p: *
\lngx_if_shape_ulc:TF * {\langle true code \rangle}{\langle false code \rangle}

```

These conditionals select the true branch if the `up`, `it`, `sc`, `ssc`, `sl`, `sw`, `ulc` shapes (respectively) are active, false otherwise.

```

\lngx_super_font_family:nn {\langle family id \rangle} {\langle rm=\langle rm nfss \rangle, sf=\langle sf nfss \rangle, tt=\langle tt nfss \rangle \rangle}

```

This function takes an $\langle id \rangle$ and sets the `rm`, `sf`, `tt` values as requested by the user and creates a super font family.

```

\lngx_soft_super_font_family:nn {\langle id \rangle}{\langle encoding, family, series, shape \rangle}
\lngx_softer_super_font_family:n {\langle id \rangle}
\lngx_softest_super_font_family:n {\langle id \rangle}

```

The `\lngx_soft_super_font_family:nn` sets super family marked by the $\langle id \rangle$ and reactivates the currently active font attributes listed in the second argument. The other two do the same, but without the list. the `softer` one omits the encoding and the `softest` one reactivate all of them.

```

\lngx_soft_normal_font:n {\langle id \rangle}

```

```

\lngx_softer_normal_font:
\lngx_softest_normal_font:

```

Quite similar to the soft super family functions, these ones set the default font family and reactivate the font attributes. The `soft` one sets the attributes listed in the argument. The `softer` one omits encoding and reactivates the rest and the `softest` one reactivates all.

Implementation

In this section the code of this bundle is documented. Each package in the bundle is documented in a separate subsection.

LINGUIS $\overline{\text{Ti}}$ X

Provide the package with its basic information.

```
1 <*package>
2 \ProvidesExplPackage{linguistix}
3     {2025-06-19}
4     {v0.5a}
5     {%
6         The ‘Linguis $\overline{\text{Ti}}$ X’ bundle: Enhanced
7         support for linguistics.%
8     }
```

When one loads LINGUIS $\overline{\text{Ti}}$ X, all the packages of the bundle are loaded automatically. That’s the only content of the umbrella package LINGUIS $\overline{\text{Ti}}$ X. All the packages are loaded conditionally (i.e., only if not loaded already).

```
9
10 \IfPackageLoadedF { linguistix-base } {
11     \RequirePackage { linguistix-base }
12 }
13 \IfPackageLoadedF { linguistix-fonts } {
14     \RequirePackage { linguistix-fonts }
15 }
16 \IfPackageLoadedF { linguistix-ipa } {
17     \RequirePackage { linguistix-ipa }
18 }
19 \IfPackageLoadedF { linguistix-logos } {
20     \RequirePackage { linguistix-logos }
21 }
22 \IfPackageLoadedF { linguistix-nfss } {
23     \RequirePackage { linguistix-nfss }
24 }
25 </package>
```

Set the essentials of the package.

```

26 <*base>
27 \ProvidesExplPackage{linguistix-base}
28             {2025-06-19}
29             {v0.5a}
30             {%
31             The base package of the ‘LinguisTiX’
32             bundle.%
33             }

```

\lngx_set_keys:n I use the `l3keys` module of L^AT_EX₃ for creating the key-values used in this bundle. In order to get a singleton parser for all the packages of the bundle, I have create this parsing command that is used throughout the bundle.

```

34
35 \cs_new_protected:Npn \lngx_set_keys:n #1 {
36   \keys_set:nn { lngx _ keys } { #1 }
37 }

```

(End of definition for \lngx_set_keys:n. This function is documented on page 12.)

\linguistix I equate this command with a user-side macro here and end the LINGUIS**T****I****X**-BASE package.

```

38
39 \cs_gset_eq:NN \linguistix \lngx_set_keys:n
40 </base>

```

(End of definition for \linguistix. This function is documented on page 5.)

The `unicode-math` defines `\gla` which clashes with the same command defined by the `expex` package. Of course, the `expex-\gla` is more relevant in linguistics. Thus I will save that and provide a new command for the `unicode-math-\gla`. This is not relevant to people who are not using `expex`. Thus, the settings are loaded only conditionally.

```

41 \fixpex
42 \ProvidesExplPackage{linguistix-fixpex}
43     {2025-06-19}
44     {v0.5a}
45     {%
46         The base package of the 'Linguistix'
47         bundle.%
48     }

```

This package is useful only if either `expex` or `unicode-math` is loaded. Otherwise, it is of no use. Thus, I create a message when either of them is not loaded.

```

49
50 \msg_new:nnn { fixpex } { pkg_not_loaded } {
51     The~ 'linguistix-fixpex'~ package~ is~ a~ first-aid~
52     for~ resolving~ the~ conflict~ between~ 'unicode-math'~
53     and~\ 'expex'.~ It~ should~ only~ be~ used~ if~ at~ least~
54     one~ of~ these~ two~ is~ loaded.~ Here~
55     'linguistix-fixpex'~ can~\ be~ omitted~ since~ you~ are~
56     not~ using~ '#1'.
57 }

```

I first start the hook `begindocument/before`.

```

58
59 \hook_gput_code:nnn { begindocument / before } { . } {

```

The `unicode-math` package defines `\gla` after `\begin{document}`, so the fix needs to be added after that is done. For that, I start the `begindocument/end` hook.

```

60     \IfPackageLoadedTF { expex } {
61         \IfPackageLoadedTF { unicode-math } {
62             \hook_gput_code:nnn { begindocument / end } { . } {

```

`\umgla` This replicates the `unicode-math-\gla` for future use.

```

63         \cs_set_eq:NN \umgla \gla

```

(End of definition for `\umgla`. This function is documented on page 5.)

The `expex-\gla` is then equated to the internal function of the package that does the actual function (Munn & Gregorio, 2023).

```

64         \cs_set_eq:NN \gla \glw@gla
65     }

```

In the false branch of `unicode-math`, I issue an info message that is not visible on the terminal, but is printed in the log file.

```

66     } {
67         \msg_info:nnn { fixpex } { pkg_not_loaded } {
68             unicode-math
69         }
70     }

```

Similarly, I do it for expex.

```
71 } {  
72   \msg_info:nnn { fixpex } { pkg_not_loaded } {  
73     expex  
74   }  
75 }  
76 }  
77 </fixpex>
```

Package essentials first.

```

78 <{*font}
79 \ProvidesExplPackage{linguistix-fonts}
80     {2025-06-19}
81     {v0.5a}
82     {%
83         The font-assistant package of the
84         'LinguisTiX' bundle.%
85     }

```

I load LINGUIS**TI**X-BASE and unicode-math (if they are not already loaded).

```

86
87 \IfPackageLoadedF { linguistix-base } {
88     \RequirePackage { linguistix-base }
89 }
90
91 \IfPackageLoadedF { unicode-math } {
92     \RequirePackage { unicode-math }
93 }
94
95 \IfPackageLoadedF { linguistix-fixpex } {
96     \RequirePackage { linguistix-fixpex }
97 }

```

\LaTeX We save the original code for the **\LaTeX** logo and then renew the command.
\ogLaTeX

```

98
99 \NewCommandCopy \ogLaTeX \LaTeX
100
101 \RenewDocumentCommand \LaTeX { } {%
102     L\kern-.81ex\relax
103     \raisebox{.6ex}{\textsc{a}}\kern-.23ex\relax
104     \hbox{T}\kern-.4ex\relax
105     \raisebox{-.5ex}{E}\kern-.3ex\relax
106     X%
107 }

```

(End of definition for **\LaTeX** and **\ogLaTeX**. These functions are documented on page 5.)

old style numbers I use the **.bool_gset:N** key-type of **l3keys** for developing these boolean keys.
\g_lngx_old_style_bool
old style one
\g_lngx_old_style_one_bool
bourbaki's empty set
\g_lngx_bourbaki_bool

```

108
109 \keys_define:nn { lngx _ keys } {
110     old~ style~ numbers
111     .bool_gset:N          = {
112         \g_lngx_old_style_bool
113     },
114     old~ style~ one
115     .bool_gset:N          = {
116         \g_lngx_old_style_one_bool
117     },
118     bourbaki's~ empty~ set
119     .bool_gset:N          = {
120         \g_lngx_bourbaki_bool
121     }

```

I22 }

(End of definition for old style numbers and others. These functions are documented on page 6.)

```

text upright
text upright features
text bold upright
text bold upright features
text italic
text italic features
text bold italic
text bold italic features
text slanted
text slanted features
text bold slanted
text bold slanted features
text swash
text swash features
text bold swash
text small caps
text small caps features
\g_lngx_text_upright_tl
\g_lngx_text_upright_features_tl
\g_lngx_text_bold_upright_tl
\g_lngx_text_bold_upright_features_tl
\g_lngx_text_italic_tl
\g_lngx_text_italic_features_tl
\g_lngx_text_bold_italic_tl
\g_lngx_text_bold_italic_features_tl
\g_lngx_text_slanted_tl
\g_lngx_text_slanted_features_tl
\g_lngx_text_bold_slanted_tl
\g_lngx_text_bold_slanted_features_tl
\g_lngx_text_swash_tl
\g_lngx_text_swash_features_tl
\g_lngx_text_bold_swash_tl
\g_lngx_text_bold_swash_features_tl
\g_lngx_text_small_caps_tl
\g_lngx_text_small_caps_features_tl

```

I save the names of the fonts in `tl` variables. This section creates the keys for serif text fonts. All these keys have a common pattern of code. For the convenience of maintenance, I have created a comma-separated-list and used the elements of this list inside the common code. (See: <https://topanswers.xyz/tex?q=8074#a7689>.)

```

123
124 \clist_map_inline:nn {
125   upright,
126   bold~ upright,
127   italic,
128   bold~ italic,
129   slanted,
130   bold~ slanted,
131   swash,
132   bold~ swash,
133   small~ caps
134 } {

```

The key-names can contain spaces, but the variables can't. I set a temporary variable and convert the spaces into underscores. Note that `#1` means the elements of the `clist` here.

```

135 \tl_set:Nn \l_tmpa_tl { #1 }
136 \tl_replace_all:Nnn \l_tmpa_tl { ~ } { _ }
137 \tl_gclear_new:c {
138   g _ lngx _ text _ \l_tmpa_tl _ features _ tl
139 }

```

All the keys here are prefixed with the word `text` in order to distinguish them from the keys provided by the `LINGUISTIX-IPA` package. The argument of these keys should be expanded for which I use `.tl_gset_e:c` type of `l3keys`.

```

140 \keys_define:nn { lngx _ keys } {
141   text~ #1
142   .tl_gset_e:c = {
143     g _ lngx _ text _ \l_tmpa_tl _ tl
144   },

```

Each of these keys is followed by its respective `features` key which is supposed to take an appending argument. The `.tl`-type keys don't support this. I create this key with the `.code:n` type. Like before, first I set a temporary variable for space-to-underscore conversion, use it with the `\tl_put_right:ce` call for appending.

```

145   text~ #1~ features
146   .code:n = {
147     \tl_set:Nn \l_tmpb_tl { #1 }
148     \tl_replace_all:Nnn \l_tmpb_tl { ~ } { _ }
149     \tl_put_right:ce {
150       g _ lngx _ text _ \l_tmpb_tl _ features _ tl
151     } { ##1 , }

```

Lastly, we clear the temporary `tl`s.

```

152   \tl_clear:N \l_tmpb_tl
153 }
154 }
155 \tl_clear:N \l_tmpa_tl
156 }

```

(End of definition for `text upright` and others. These functions are documented on page 8.)

```

text sans upright
text sans upright features
text sans bold upright
text sans bold upright features
text sans italic
text sans italic features
text sans bold italic
text sans bold italic features
text sans slanted
text sans slanted features
text sans bold slanted
text sans bold slanted features
text sans swash
text sans swash features
text sans bold swash
text sans bold swash features
text sans small caps
text sans small caps features
\g_lngx_text_sans_upright_tl
\g_lngx_text_sans_upright_features_tl
\g_lngx_text_sans_bold_upright_tl
\g_lngx_text_sans_bold_upright_features_tl
\g_lngx_text_sans_italic_tl
\g_lngx_text_sans_italic_features_tl
\g_lngx_text_sans_bold_italic_tl
\g_lngx_text_sans_bold_italic_features_tl
\g_lngx_text_sans_slanted_tl
\g_lngx_text_sans_slanted_features_tl
\g_lngx_text_sans_bold_slanted_tl
\g_lngx_text_sans_bold_slanted_features_tl
\g_lngx_text_sans_swash_tl
\g_lngx_text_sans_swash_features_tl
\g_lngx_text_sans_bold_swash_tl
\g_lngx_text_sans_bold_swash_features_tl
\g_lngx_text_sans_small_caps_tl
\g_lngx_text_sans_small_caps_features_tl

```

With this same mechanism, the keys for sans serif fonts are developed.

```

157
158 \clist_map_inline:nn {
159   upright,
160   bold~ upright,
161   italic,
162   bold~ italic,
163   slanted,
164   bold~ slanted,
165   swash,
166   bold~ swash,
167   small~ caps
168 } {
169   \tl_set:Nn \l_tmpa_tl { #1 }
170   \tl_replace_all:Nnn \l_tmpa_tl { ~ } { _ }
171   \tl_gclear_new:c {
172     g _ lngx _ text _ sans _ \l_tmpa_tl _ features _ tl
173   }
174   \keys_define:nn { lngx _ keys } {
175     text~ sans~ #1
176     .tl_gset_e:c          = {
177       g _ lngx _ text _ sans _ \l_tmpa_tl _ tl
178     },
179     text~ sans~ #1~ features
180     .code:n              = {
181       \tl_set:Nn \l_tmpb_tl { #1 }
182       \tl_replace_all:Nnn \l_tmpb_tl { ~ } { _ }
183       \tl_put_right:ce {
184         g _ lngx _ text _ sans _ \l_tmpb_tl _ features _ tl
185       } { ##1 , }
186       \tl_clear:N \l_tmpb_tl
187     }
188   }
189   \tl_clear:N \l_tmpa_tl
190 }

```

(End of definition for *text sans upright* and others. These functions are documented on page 8.)

Here, with the same setup, I develop the keys for monospaced fonts.

```

text mono upright
text mono upright features
text mono bold upright
text mono bold upright features
text mono italic
text mono italic features
text mono bold italic
text mono bold italic features
text mono slanted
text mono slanted features
text mono bold slanted
text mono bold slanted features
text mono swash
text mono swash features
text mono bold swash
text mono bold swash features
text mono small caps
text mono small caps features
\g_lngx_text_mono_upright_tl
\g_lngx_text_mono_upright_features_tl
\g_lngx_text_mono_bold_upright_tl
\g_lngx_text_mono_bold_upright_features_tl
\g_lngx_text_mono_italic_tl
\g_lngx_text_mono_italic_features_tl
\g_lngx_text_mono_bold_italic_tl
\g_lngx_text_mono_bold_italic_features_tl
\g_lngx_text_mono_slanted_tl
\g_lngx_text_mono_slanted_features_tl
\g_lngx_text_mono_bold_slanted_tl
\g_lngx_text_mono_bold_slanted_features_tl
\g_lngx_text_mono_swash_tl
\g_lngx_text_mono_swash_features_tl
\g_lngx_text_mono_bold_swash_tl
\g_lngx_text_mono_bold_swash_features_tl
\g_lngx_text_mono_small_caps_tl
\g_lngx_text_mono_small_caps_features_tl

```

```

191
192 \clist_map_inline:nn {
193   upright,
194   bold~ upright,
195   italic,
196   bold~ italic,
197   slanted,
198   bold~ slanted,
199   swash,
200   bold~ swash,
201   small~ caps
202 } {
203   \tl_set:Nn \l_tmpa_tl { #1 }
204   \tl_replace_all:Nnn \l_tmpa_tl { ~ } { _ }
205   \tl_gclear_new:c {
206     g _ lngx _ text _ mono _ \l_tmpa_tl _ features _ tl
207   }
208   \keys_define:nn { lngx _ keys } {
209     text~ mono~ #1
210     .tl_gset_e:c          = {
211       g _ lngx _ text _ mono _ \l_tmpa_tl _ tl
212     },
213     text~ mono~ #1~ features
214     .code:n              = {
215       \tl_set:Nn \l_tmpb_tl { #1 }
216       \tl_replace_all:Nnn \l_tmpb_tl { ~ } { _ }
217       \tl_put_right:ce {
218         g _ lngx _ text _ mono _ \l_tmpb_tl _ features _ tl
219       } { ##1 , }
220       \tl_clear:N \l_tmpb_tl
221     }
222   }
223   \tl_clear:N \l_tmpa_tl
224 }

```

(End of definition for *text mono upright* and others. These functions are documented on page 8.)

math The following are the keys set for math. They use the same mechanism as before.

math features

math bold

math bold features

```

225
226 \keys_define:nn { lngx _ keys } {
227   math
228   .tl_gset_e:c          = {
229     g _ lngx _ math _ tl
230   },
231   math~ features
232   .tl_gset_e:c          = {
233     g _ lngx _ math _ features _ tl
234   },
235   math~ bold
236   .tl_gset_e:c          = {
237     g _ lngx _ math _ bold _ tl
238   },
239   math~ bold~ features
240   .code:n               = {
241     \tl_put_right:ce {
242       g _ lngx _ math _ bold _ features _ tl
243     } { #1 }
244   }
245 }

```

(End of definition for *math* and others. These functions are documented on page 6.)

newcm This key is of type `.meta:n`. It sets certain other keys that enable the New Computer Modern fonts in all serif, serif and monospaced families.

```

246
247 \keys_define:nn { lngx _ keys } {
248   newcm
249   .meta:n               = {
250     text~
251     upright              = {
252       NewCM10-Book.otf
253     },
254     text~
255     bold~ upright        = {
256       NewCM10-Bold.otf
257     },
258     text~
259     italic               = {
260       NewCM10-BookItalic.otf
261     },
262     text~
263     bold~ italic         = {
264       NewCM10-BoldItalic.otf
265     },
266     math
267     NewCMMath-Book.otf
268   },
269     math~ bold          = {
270       NewCMMath-Bold.otf
271     },
272     text~

```

```

273 sans~ upright          = {
274     NewCMSans10-Book.otf
275 },
276 text~
277 sans~ bold~ upright     = {
278     NewCMSans10-Bold.otf
279 },
280 text~
281 sans~ italic            = {
282     NewCMSans10-BookOblique.otf
283 },
284 text~
285 sans~ bold~ italic      = {
286     NewCMSans10-BoldOblique.otf
287 },
288 text~
289 mono~ upright           = {
290     NewCMMono10-Book.otf
291 },
292 text~
293 mono~ bold~ upright     = {
294     NewCMMono10-Bold.otf
295 },
296 text~
297 mono~ italic            = {
298     NewCMMono10-BookItalic.otf
299 },
300 text~
301 mono~ bold~ italic      = {
302     NewCMMono10-BoldOblique.otf
303 }
304 }
305 }

```

(End of definition for *newcm*. This function is documented on page 6.)

newcm sans This is a `.meta:n` key that sets the default fonts to the sans family.

```

306
307 \keys_define:nn { lngx _ keys } {
308     newcm~ sans
309     .meta:n          = {
310         text~
311         upright       = {
312             NewCMSans10-Book.otf
313         },
314         text~
315         bold upright   = {
316             NewCMSans10-Bold.otf
317         },
318         text~
319         italic         = {
320             NewCMSans10-BookOblique.otf
321         },
322         text~

```

```

323     bold~ italic          = {
324         NewCMSans10-BoldOblique.otf
325     }
326 }
327 }

```

(End of definition for *newcm sans*. This function is documented on page 6.)

newcm mono This is a `.meta:n` key that sets the default fonts to the monospaced family.

```

328
329 \keys_define:nn { lngx _ keys } {
330     newcm~ mono
331     .meta:n          = {
332         text~
333         upright       = {
334             NewCMMono10-Book.otf
335         },
336         text~
337         bold upright  = {
338             NewCMMono10-Bold.otf
339         },
340         text~
341         italic        = {
342             NewCMMono10-BookItalic.otf
343         },
344         text~
345         bold~ italic  = {
346             NewCMMono10-BoldOblique.otf
347         }
348     }
349 }

```

(End of definition for *newcm mono*. This function is documented on page 6.)

newcm regular This is a `.meta:n` key that sets the default fonts to the regular variant of the New Computer Modern family.

```

350
351 \keys_define:nn { lngx _ keys } {
352     newcm~ regular
353     .meta:n          = {
354         text~
355         upright       = {
356             NewCM10-Regular.otf
357         },
358         text~
359         bold~ upright  = {
360             NewCM10-Bold.otf
361         },
362         text~
363         italic        = {
364             NewCM10-Italic.otf
365         },
366         text~
367         bold~ italic  = {

```

```

368     NewCM10-BoldItalic.otf
369 },
370 math                    = {
371     NewCMMath-Regular.otf
372 },
373 math~ bold             = {
374     NewCMMath-Bold.otf
375 },
376 text~
377 sans~ upright          = {
378     NewCMSans10-Regular.otf
379 },
380 text~
381 sans~ bold~ upright    = {
382     NewCMSans10-Bold.otf
383 },
384 text~
385 sans~ italic           = {
386     NewCMSans10-Oblique.otf
387 },
388 text~
389 sans~ bold~ italic     = {
390     NewCMSans10-BoldOblique.otf
391 },
392 text~
393 mono~ upright          = {
394     NewCMMono10-Regular.otf
395 },
396 text~
397 mono~ bold~ upright    = {
398     NewCMMono10-Bold.otf
399 },
400 text~
401 mono~ italic           = {
402     NewCMMono10-Italic.otf
403 },
404 text~
405 mono~ bold~ italic     = {
406     NewCMMono10-Bold.otf
407 }
408 }
409 }

```

(End of definition for newcm regular. This function is documented on page 6.)

newcm regular sans This is a `.meta:n` key that sets the default fonts to the regular sans variant of the New Computer Modern family.

```

410
411 \keys_define:nn { lngx _ keys } {
412   newcm~ regular~ sans
413   .meta:n                = {
414     text~
415     upright               = {
416       NewCMSans10-Regular.otf

```

```

417 },
418 text~
419 bold~ upright          = {
420   NewCMSans10-Bold.otf
421 },
422 text~
423 italic                 = {
424   NewCMSans10-Oblique.otf
425 },
426 text~
427 bold~ italic           = {
428   NewCMSans10-BoldOblique.otf
429 }
430 }
431 }

```

(End of definition for *newcm regular sans*. This function is documented on page 6.)

newcm regular mono This is a `.meta:n` key that sets the default fonts to the regular monospaced variant of the New Computer Modern family.

```

432
433 \keys_define:nn { lngx _ keys } {
434   newcm~ regular~ mono
435   .meta:n          = {
436     text~
437     upright         = {
438       NewCMMono10-Regular.otf
439     },
440     text~
441     bold~ upright   = {
442       NewCMMono10-Bold.otf
443     },
444     text~
445     italic          = {
446       NewCMMono10-Italic.otf
447     },
448     text~
449     bold~ italic    = {
450       NewCMMono10-Bold.otf
451     }
452   }
453 }

```

(End of definition for *newcm regular mono*. This function is documented on page 6.)

By default, we load the **newcm** key that loads all the New Computer Modern fonts in its book variant.

```

454
455 \lngx_set_keys:n {
456   newcm,

```

Then we load the **bourbaki's empty set** boolean. This gets read later while setting the math font.

```

457   bourbaki's~ empty~ set,

```

Lastly we load the old `style numbers` boolean.

```

458   old~ style~ numbers
459 }

```

We need HarfBuzz renderer whenever Lua_{La}T_EX is used. For that we add the required feature to the feature-lists of all the fonts.

```

460
461 \sys_if_engine luatex:T {
462   \lngx_set_keys:n {
463     text~
464     upright~ features      = {
465       Renderer             = { HarfBuzz }
466     },
467     text~ sans~
468     upright~ features      = {
469       Renderer             = { HarfBuzz }
470     },
471     text~ mono~
472     upright~ features      = {
473       Renderer             = { HarfBuzz }
474     }
475   }
476 }

```

`\lngx_set_main_font:nn` Since I use many conditionals and values while setting the fonts, here, I develop a few wrappers around the font commands. The `\cs_generate_variant:Nn` line comes in handy to generate the argument-expanding versions of the default wrapper-commands.

```

\lngx_set_sans_font:nn
\lngx_set_mono_font:nn
\lngx_set_math_font:nn
477
478 \cs_new_protected:Npn \lngx_set_main_font:nn #1#2 {
479   \setmainfont [ #1 ] { #2 }
480 }
481
482 \cs_new_protected:Npn \lngx_set_sans_font:nn #1#2 {
483   \setsansfont [ #1 ] { #2 }
484 }
485
486 \cs_new_protected:Npn \lngx_set_mono_font:nn #1#2 {
487   \setmonofont [ #1 ] { #2 }
488 }
489
490 \cs_new_protected:Npn \lngx_set_math_font:nn #1#2 {
491   \setmathfont [ #1 ] { #2 }
492 }
493
494 \cs_generate_variant:Nn \lngx_set_main_font:nn { ee }
495 \cs_generate_variant:Nn \lngx_set_sans_font:nn { ee }
496 \cs_generate_variant:Nn \lngx_set_mono_font:nn { ee }
497 \cs_generate_variant:Nn \lngx_set_math_font:nn { ee }

```

(End of definition for `\lngx_set_main_font:nn` and others. These functions are documented on page 19.)

Now I start the pre-begindocument hook. New Computer Modern comes in two sizes for some shapes, 8 and 10. They matter for micro-typographic perfection. I have a little complicated checking for providing support for the entire New Computer Modern family.

First I check if the font that is set to be the main font is New Computer Modern or not. For that, searching for the keyword `NewCM` suffices.

```

498
499 \hook_gput_code:nnn { begindocument / before } { . } {
500   \tl_if_in:cnT {
501     g _ lngx _ text _ upright _ tl
502   } { NewCM } {

```

The Book weight of New Computer Modern consistently has `Book` in all its font-file-names. I test over that to distinguish it from the regular weight. In the true branch of it, I add the size features as required by fontspec for setting size-specific fonts.

```

503   \tl_if_in:cnTF {
504     g _ lngx _ text _ upright _ tl
505   } { Book } {
506     \lngx_set_keys:n {
507       text~
508       upright~ features      = {
509         SizeFeatures         = {
510           {
511             Size              = {-8},
512             Font              = {
513               NewCM08-Book.otf
514             }
515           },
516           {
517             Size              = {8-},
518             Font              = {
519               NewCM10-Book.otf
520             }
521           }
522         }
523       }
524     }

```

In the false branch, the same settings are used for the regular variant.

```

525   } {
526     \lngx_set_keys:n {
527       text~
528       upright~ features      = {
529         SizeFeatures         = {
530           {
531             Size              = {-8},
532             Font              = {
533               NewCM08-Regular.otf
534             }
535           },
536           {
537             Size              = {8-},
538             Font              = {
539               NewCM10-Regular.otf
540             }
541           }
542         }
543       }

```



```

544     }
545   }
546 }

```

When the `newcm sans` key is loaded, sans fonts are set as main fonts. All the sans variants have `NewCMSans` in their file-names. I repeat the same check for this case. This is on purpose loaded later, so that the features loaded by the previous snippet are overridden by this one in case the main font is sans².

```

547 \tl_if_in:cnT {
548   g _ lngx _ text _ upright _ tl
549 } { NewCMSans } {
550   \tl_if_in:cnTF {
551     g _ lngx _ text _ upright _ tl
552   } { Book } {
553     \lngx_set_keys:n {
554       text~
555       upright~ features = {
556         SizeFeatures = {
557           {
558             Size = {-8},
559             Font = {
560               NewCMSans08-Book.otf
561             }
562           },
563           {
564             Size = {8-},
565             Font = {
566               NewCMSans10-Book.otf
567             }
568           }
569         }
570       }
571     }
572   } {
573     \lngx_set_keys:n {
574       text~
575       upright~ features = {
576         SizeFeatures = {
577           {
578             Size = {-8},
579             Font = {
580               NewCMSans08-Regular.otf
581             }
582           },
583           {
584             Size = {8-},
585             Font = {
586               NewCMSans10-Regular.otf
587             }
588           }
589         }
590       }

```

²The test for `NewCM` matches with fonts that have `NewCMSans` too and this is the fastest test I could think of. Suggestions for alternative methods are highly welcome.

```

591     }
592   }
593 }

```

Italic fonts also have this size variant, so here we repeat the same checks for Italic.

```

594 \tl_if_in:cnT {
595   g _ lngx _ text _ italic _ tl
596 } { NewCM } {
597   \tl_if_in:cnTF {
598     g _ lngx _ text _ italic _ tl
599   } { Book } {
600     \lngx_set_keys:n {
601       text~
602       italic~ features      = {
603         SizeFeatures        = {
604           {
605             Size              = {-8},
606             Font              = {
607               NewCM08-BookItalic.otf
608             }
609           },
610           {
611             Size              = {8-},
612             Font              = {
613               NewCM10-BookItalic.otf
614             }
615           }
616         }
617       }
618     }
619   } {
620     \lngx_set_keys:n {
621       text~
622       italic~ features      = {
623         SizeFeatures        = {
624           {
625             Size              = {-8},
626             Font              = {
627               NewCM08-Italic.otf
628             }
629           },
630           {
631             Size              = {8-},
632             Font              = {
633               NewCM10-Italic.otf
634             }
635           }
636         }
637       }
638     }
639   }
640 }
641 \tl_if_in:cnT {
642   g _ lngx _ text _ italic _ tl
643 } { NewCMSans } {

```

```

644 \tl_if_in:cnTF {
645   g _ lngx _ text _ italic _ tl
646 } { Book } {
647   \lngx_set_keys:n {
648     text~
649     italic~ features      = {
650       SizeFeatures        = {
651         {
652           Size              = {-8},
653           Font              = {
654             NewCMSans08-BookOblique.otf
655           }
656         },
657         {
658           Size              = {8-},
659           Font              = {
660             NewCMSans10-BookOblique.otf
661           }
662         }
663       }
664     }
665   }
666 } {
667   \lngx_set_keys:n {
668     text~
669     italic~ features      = {
670       SizeFeatures        = {
671         {
672           Size              = {-8},
673           Font              = {
674             NewCMSans08-Oblique.otf
675           }
676         },
677         {
678           Size              = {8-},
679           Font              = {
680             NewCMSans10-Oblique.otf
681           }
682         }
683       }
684     }
685   }
686 }
687 }

```

By default, I have set sans fonts from this family in a different set of variables. I repeat the same checks again for those variables. These coexist with the serif variables.

```

688 \tl_if_in:cnT {
689   g _ lngx _ text _ sans _ upright _ tl
690 } { NewCMSans } {
691   \tl_if_in:cnTF {
692     g _ lngx _ text _ upright _ tl
693   } { Book } {
694     \lngx_set_keys:n {

```

```

695     text~ sans~
696     upright~ features      = {
697         SizeFeatures      = {
698             {
699                 Size        = {-8},
700                 Font        = {
701                     NewCMSans08-Book.otf
702                 }
703             },
704             {
705                 Size        = {8-},
706                 Font        = {
707                     NewCMSans10-Book.otf
708                 }
709             }
710         }
711     }
712 }
713 } {
714     \ltx_set_keys:n {
715         text~ sans~
716         upright~ features      = {
717             SizeFeatures      = {
718                 {
719                     Size        = {-8},
720                     Font        = {
721                         NewCMSans08-Regular.otf
722                     }
723                 },
724                 {
725                     Size        = {8-},
726                     Font        = {
727                         NewCMSans10-Regular.otf
728                     }
729                 }
730             }
731         }
732     }
733 }
734 }
735 \tl_if_in:cnT {
736     g _ lngx _ text _ sans _ italic _ tl
737 } { NewCMSans } {
738     \tl_if_in:cnTF {
739         g _ lngx _ text _ italic _ tl
740     } { Book } {
741         \ltx_set_keys:n {
742             text~ sans~
743             italic~ features      = {
744                 SizeFeatures      = {
745                     {
746                         Size        = {-8},
747                         Font        = {
748                             NewCMSans08-BookOblique.otf

```

```

749         }
750     },
751     {
752         Size = {8-},
753         Font = {
754             NewCMSans10-BookOblique.otf
755         }
756     }
757 }
758 }
759 }
760 } {
761     \lngx_set_keys:n {
762         text~ sans~
763         italic~ features = {
764             SizeFeatures = {
765                 {
766                     Size = {-8},
767                     Font = {
768                         NewCMSans08-Oblique.otf
769                     }
770                 },
771                 {
772                     Size = {8-},
773                     Font = {
774                         NewCMSans10-Oblique.otf
775                     }
776                 }
777             }
778         }
779     }
780 }
781 }

```

Now I load the fonts and features. I am using variables that need to be loaded at the end so that all the intermediate user-given changes are also read and considered. Every sub-font (e.g., bold font, Italic font) is stored in a `tl`. Here I save the features as required by `fontspec` in LINGUISTIC feature keys.

```

782 \lngx_set_keys:n {
783     text~
784     upright~ features = {
785         UprightFont = {
786             \g_lngx_text_upright_tl
787         },
788         UprightFeatures = {
789             \g_lngx_text_upright_features_tl
790         },
791         ItalicFont = {
792             \g_lngx_text_italic_tl
793         },
794         ItalicFeatures = {
795             \g_lngx_text_italic_features_tl
796         },
797         BoldFont = {

```

```

798     \g_lngx_text_bold_upright_tl
799 },
800 BoldFeatures          = {
801     \g_lngx_text_bold_upright_features_tl
802 },
803 BoldItalicFont        = {
804     \g_lngx_text_bold_italic_tl
805 },
806 BoldItalicFeatures    = {
807     \g_lngx_text_bold_italic_features_tl
808 },

```

The New Computer Modern fonts don't have the following shapes, but other fonts may have them, so I load the variables conditionally (i.e., only if they are not empty).

```

809     \tl_if_empty:cF {
810         g _ lngx _ text _ slanted _ tl
811     } {
812         SlantedFont      = {
813             \g_lngx_text_slanted_tl
814         },
815         \tl_if_empty:cF {
816             g _ lngx _ text _ slanted _ features _ tl
817         } {
818             SlantedFeatures = {
819                 \g_lngx_text_slanted_features_tl
820             },
821         }
822     }
823     \tl_if_empty:cF {
824         g _ lngx _ text _ bold _ slanted _ tl
825     } {
826         BoldSlantedFont  = {
827             \g_lngx_text_bold_slanted_tl
828         },
829         BoldSlantedFeatures = {
830             \g_lngx_text_bold_slanted_features_tl
831         },
832     }
833     \tl_if_empty:cF {
834         g _ lngx _ text _ swash _ tl
835     } {
836         SwashFont        = {
837             \g_lngx_text_swash_tl
838         },
839         SwashFeatures     = {
840             \g_lngx_text_swash_features_tl
841         },
842     }
843     \tl_if_empty:cF {
844         g _ lngx _ text _ bold _ swash _ tl
845     } {
846         BoldSwashFont     = {
847             \g_lngx_text_bold_swash_tl
848         },

```

```

849         BoldSwashFeatures      = {
850             \g_lngx_text_bold_swash_features_tl
851         },
852     }
853     \tl_if_empty:cF {
854         g _ lngx _ text _ small _ caps _ tl
855     } {
856         SmallCapsFont           = {
857             \g_lngx_text_small_caps_tl
858         },
859         SmallCapsFeatures       = {
860             \g_lngx_text_small_caps_features_tl
861         }
862     }
863 },

```

Exactly like serif fonts, I develop the feature-set for sans and mono fonts.

```

864 text~ sans~
865 upright~ features      = {
866     UprightFont          = {
867         \g_lngx_text_sans_upright_tl
868     },
869     UprightFeatures      = {
870         \g_lngx_text_sans_upright_features_tl
871     },
872     BoldFont             = {
873         \g_lngx_text_sans_bold_upright_tl
874     },
875     BoldFeatures         = {
876         \g_lngx_text_sans_bold_upright_features_tl
877     },
878     ItalicFont           = {
879         \g_lngx_text_sans_italic_tl
880     },
881     ItalicFeatures       = {
882         \g_lngx_text_sans_italic_features_tl
883     },
884     BoldItalicFont       = {
885         \g_lngx_text_sans_bold_italic_tl
886     },
887     BoldItalicFeatures   = {
888         \g_lngx_text_sans_bold_italic_features_tl
889     },
890     \tl_if_empty:cF {
891         g _ lngx _ text _ sans _ slanted _ tl
892     } {
893         SlantedFont       = {
894             \g_lngx_text_sans_slanted_tl
895         },
896         SlantedFeatures    = {
897             \g_lngx_text_sans_slanted_features_tl
898         },
899     }
900     \tl_if_empty:cF {
901         g _ lngx _ text _ sans _ bold _ slanted _ tl

```

```

902 } {
903     BoldSlantedFont      = {
904         \g_lngx_text_sans_bold_slanted_tl
905     },
906     BoldSlantedFeatures  = {
907         \g_lngx_text_sans_bold_slanted_features_tl
908     },
909 }
910 \tl_if_empty:cF {
911     g _ lngx _ text _ sans _ swash _ tl
912 } {
913     SwashFont            = {
914         \g_lngx_text_sans_swash_tl
915     },
916     SwashFeatures        = {
917         \g_lngx_text_sans_swash_features_tl
918     },
919 }
920 \tl_if_empty:cF {
921     g _ lngx _ text _ sans _ bold _ swash _ tl
922 } {
923     BoldSwashFont        = {
924         \g_lngx_text_sans_bold_swash_tl
925     },
926     BoldSwashFeatures    = {
927         \g_lngx_text_sans_bold_swash_features_tl
928     },
929 }
930 \tl_if_empty:cF {
931     g _ lngx _ text _ sans _ small _ caps _ tl
932 } {
933     SmallCapsFont        = {
934         \g_lngx_text_sans_small_caps_tl
935     },
936     SmallCapsFeatures    = {
937         \g_lngx_text_sans_small_caps_features_tl
938     }
939 }
940 },
941 text~ mono~
942 upright~ features      = {
943     UprightFont          = {
944         \g_lngx_text_mono_upright_tl
945     },
946     UprightFeatures      = {
947         \g_lngx_text_mono_upright_features_tl
948     },
949     BoldFont             = {
950         \g_lngx_text_mono_bold_upright_tl
951     },
952     BoldFeatures         = {
953         \g_lngx_text_mono_bold_upright_features_tl
954     },
955     ItalicFont           = {

```



```

956     \g_lngx_text_mono_italic_tl
957 },
958 ItalicFeatures          = {
959     \g_lngx_text_mono_italic_features_tl
960 },
961 BoldItalicFont          = {
962     \g_lngx_text_mono_bold_italic_tl
963 },
964 BoldItalicFeatures      = {
965     \g_lngx_text_mono_bold_italic_features_tl
966 },
967 \tl_if_empty:cF {
968     g _ lngx _ text _ mono _ slanted _ tl
969 } {
970     SlantedFont          = {
971         \g_lngx_text_mono_slanted_tl
972     },
973     SlantedFeatures      = {
974         \g_lngx_text_mono_slanted_features_tl
975     },
976 }
977 \tl_if_empty:cF {
978     g _ lngx _ text _ mono _ bold _ slanted _ tl
979 } {
980     BoldSlantedFont      = {
981         \g_lngx_text_mono_bold_slanted_tl
982     },
983     BoldSlantedFeatures  = {
984         \g_lngx_text_mono_bold_slanted_features_tl
985     },
986 }
987 \tl_if_empty:cF {
988     g _ lngx _ text _ mono _ swash _ tl
989 } {
990     SwashFont            = {
991         \g_lngx_text_mono_swash_tl
992     },
993     SwashFeatures        = {
994         \g_lngx_text_mono_swash_features_tl
995     },
996 }
997 \tl_if_empty:cF {
998     g _ lngx _ text _ mono _ bold _ swash _ tl
999 } {
1000     BoldSwashFont        = {
1001         \g_lngx_text_mono_bold_swash_tl
1002     },
1003     BoldSwashFeatures    = {
1004         \g_lngx_text_mono_bold_swash_features_tl
1005     },
1006 }
1007 \tl_if_empty:cF {
1008     g _ lngx _ text _ mono _ small _ caps _ tl
1009 } {

```

```

1010     SmallCapsFont          = {
1011       \g_lngx_text_mono_small_caps_tl
1012     },
1013     SmallCapsFeatures       = {
1014       \g_lngx_text_mono_small_caps_features_tl
1015     }
1016   }
1017 }
1018 }
1019 \bool_if:NT \g_lngx_old_style_bool {
1020   \lngx_set_keys:n {
1021     text~
1022     upright~ features      = {
1023       Numbers              = { OldStyle }
1024     },
1025     text~ sans~
1026     upright~ features      = {
1027       Numbers              = { OldStyle }
1028     }
1029   }
1030   \tl_if_in:cnT {
1031     g_lngx_math_tl
1032   } { NewCM } {
1033     \bool_if:NT \g_lngx_old_style_one_bool {
1034       \lngx_set_keys:n {
1035         text~
1036         upright~ features  = {
1037           CharacterVariant = { 6 }
1038         },
1039         text~ sans~
1040         upright~ features  = {
1041           CharacterVariant = { 6 }
1042         }
1043       }
1044     }
1045   }
1046 }
1047 \tl_if_in:cnT {
1048   g _ lngx _ math _ tl
1049 } { NewCM } {
1050   \bool_if:NT \g_lngx_bourbaki_bool {
1051     \lngx_set_keys:n {
1052       math~ features      = {
1053         CharacterVariant   = { 1 }
1054       }
1055     }
1056   }
1057 }

```

If the New Computer Modern fonts are used, we don't need their .fontspec files as I already have incorporated all their settings in the package itself. So I have used the IgnoreFontspecFile option for fontspec.

```

1058   \tl_if_in:cnT {
1059     g _ lngx _ text _ upright _ tl

```

```

1060 } { NewCM } {
1061   \lngx_set_keys:n {
1062     text~
1063     upright~ features      = {
1064       IgnoreFontspecFile
1065     }
1066   }
1067 }
1068 \tl_if_in:cnT {
1069   g _ lngx _ text _ sans _ upright _ tl
1070 } { NewCM } {
1071   \lngx_set_keys:n {
1072     text~
1073     sans~ upright~ features = {
1074       IgnoreFontspecFile
1075     }
1076   }
1077 }
1078 \tl_if_in:cnT {
1079   g _ lngx _ text _ mono _ upright _ tl
1080 } { NewCM } {
1081   \lngx_set_keys:n {
1082     text~
1083     mono~ upright~ features = {
1084       IgnoreFontspecFile
1085     }
1086   }
1087 }
1088 \lngx_set_main_font:ee {
1089   \g_lngx_text_upright_features_tl
1090 } {
1091   \g_lngx_text_upright_tl
1092 }
1093 \lngx_set_sans_font:ee {
1094   \g_lngx_text_sans_upright_features_tl
1095 } {
1096   \g_lngx_text_sans_upright_tl
1097 }
1098 \lngx_set_mono_font:ee {
1099   \g_lngx_text_mono_upright_features_tl
1100 } {
1101   \g_lngx_text_mono_upright_tl
1102 }
1103 \lngx_set_math_font:ee {
1104   \g_lngx_math_features_tl
1105 } {
1106   \g_lngx_math_tl
1107 }
1108 }
1109 </font>

```

```

III0 <*ipa>
III1 \ProvidesExplPackage{linguistix-ipa}
III2     {2025-06-19}
III3     {v0.5a}
III4     {%
III5         A package for typesetting the IPA
III6         (International Phonetic Alphabet) from
III7         the ‘LinguisTiX’ bundle.%
III8     }

```

Then, I load unicode-math, LINGUISTIX-NFSS and LINGUISTIX-BASE (if they are not already loaded).

```

III9
III10 \IfPackageLoadedF { unicode-math } {
III11     \RequirePackage { unicode-math }
III12 }
III13
III14 \IfPackageLoadedF { linguistix-base } {
III15     \RequirePackage { linguistix-base }
III16 }
III17
III18 \IfPackageLoadedF { linguistix-nfss } {
III19     \RequirePackage { linguistix-nfss }
III20 }
III21
III22 \IfPackageLoadedF { linguistix-fixpex } {
III23     \RequirePackage { linguistix-fixpex }
III24 }

```

\ipatext The \ipatext command along with its starred variant is developed here.
\ipatext*

```

III25
III26 \NewDocumentCommand \ipatext { s m } {
III27     \IfBooleanTF { #1 } {
III28         {
III29             \lngxipa
III30             / #2 /
III31         }
III32     } {
III33         {
III34             \lngxipa
III35             [ #2 ]
III36         }
III37     }
III38 }

```

(End of definition for \ipatext and \ipatext. These functions are documented on page 7.)*

These variables store the values for fonts and features for the serif IPA.

```

ipa upright
ipa upright features
ipa bold upright
ipa bold upright features
ipa italic
ipa italic features
ipa bold italic
ipa bold italic features
ipa slanted
ipa slanted features
ipa bold slanted
ipa bold slanted features
ipa swash
ipa swash features
ipa bold swash
ipa bold swash features
ipa small caps
ipa small caps features
\g_lngx_ipa_upright_tl
\g_lngx_ipa_upright_features_tl
\g_lngx_ipa_bold_upright_tl
\g_lngx_ipa_bold_upright_features_tl
\g_lngx_ipa_italic_tl
\g_lngx_ipa_italic_features_tl
\g_lngx_ipa_bold_italic_tl
\g_lngx_ipa_bold_italic_features_tl
\g_lngx_ipa_slanted_tl
\g_lngx_ipa_slanted_features_tl
\g_lngx_ipa_bold_slanted_tl
\g_lngx_ipa_bold_slanted_features_tl
\g_lngx_ipa_swash_tl
\g_lngx_ipa_swash_features_tl
\g_lngx_ipa_bold_swash_tl
\g_lngx_ipa_bold_swash_features_tl
\g_lngx_ipa_small_caps_tl
\g_lngx_ipa_small_caps_features_tl

```

```

II49
II50 \clist_map_inline:nn {
II51   upright,
II52   bold~ upright,
II53   italic,
II54   bold~ italic,
II55   slanted,
II56   bold~ slanted,
II57   swash,
II58   bold~ swash,
II59   small~ caps
II60 } {
II61   \tl_set:Nn \l_tmpa_tl { #1 }
II62   \tl_replace_all:Nnn \l_tmpa_tl { ~ } { _ }
II63   \tl_gclear_new:c {
II64     g _ lngx _ ipa _ \l_tmpa_tl _ features _ tl
II65   }
II66   \keys_define:nn { lngx _ keys } {
II67     ipa~ #1
II68     .tl_gset_e:c          = {
II69       g _ lngx _ ipa _ \l_tmpa_tl _ tl
II70     },
II71     ipa~ #1~ features
II72     .code:n              = {
II73       \tl_set:Nn \l_tmpb_tl { #1 }
II74       \tl_replace_all:Nnn \l_tmpb_tl { ~ } { _ }
II75       \tl_put_right:ce {
II76         g _ lngx _ ipa _ \l_tmpb_tl _ features _ tl
II77       } { ##1 , }
II78       \tl_clear:N \l_tmpb_tl
II79     }
II80   }
II81   \tl_clear:N \l_tmpa_tl
II82 }

```

(End of definition for *ipa upright* and others. These functions are documented on page 8.)

These variables store the values for fonts and features for the sans IPA.

```

ipa sans upright
ipa sans upright features
ipa sans bold upright
ipa sans bold upright features
ipa sans italic
ipa sans italic features
ipa sans bold italic
ipa sans bold italic features
ipa sans slanted
ipa sans slanted features
ipa sans bold slanted
ipa sans bold slanted features
ipa sans swash
ipa sans swash features
ipa sans bold swash
ipa sans bold swash features
ipa sans small caps
ipa sans small caps features
\g_lngx_ipa_sans_upright_tl
\g_lngx_ipa_sans_upright_features_tl
\g_lngx_ipa_sans_bold_upright_tl
\g_lngx_ipa_sans_bold_upright_features_tl
\g_lngx_ipa_sans_italic_tl
\g_lngx_ipa_sans_italic_features_tl
\g_lngx_ipa_sans_bold_italic_tl
\g_lngx_ipa_sans_bold_italic_features_tl
\g_lngx_ipa_sans_slanted_tl
\g_lngx_ipa_sans_slanted_features_tl
\g_lngx_ipa_sans_bold_slanted_tl
\g_lngx_ipa_sans_bold_slanted_features_tl
\g_lngx_ipa_sans_swash_tl
\g_lngx_ipa_sans_swash_features_tl
\g_lngx_ipa_sans_bold_swash_tl
\g_lngx_ipa_sans_bold_swash_features_tl
\g_lngx_ipa_sans_small_caps_tl
\g_lngx_ipa_sans_small_caps_features_tl

```

```

II83
II84 \clist_map_inline:nn {
II85   upright,
II86   bold~ upright,
II87   italic,
II88   bold~ italic,
II89   slanted,
II90   bold~ slanted,
II91   swash,
II92   bold~ swash,
II93   small~ caps
II94 } {
II95   \tl_set:Nn \l_tmpa_tl { #1 }
II96   \tl_replace_all:Nnn \l_tmpa_tl { ~ } { _ }
II97   \tl_gclear_new:c {
II98     g _ lngx _ ipa _ mono _ \l_tmpa_tl _ features _ tl
II99   }
I200   \keys_define:nn { lngx _ keys } {
I201     ipa~ mono~ #1
I202     .tl_gset_e:c          = {
I203       g _ lngx _ ipa _ mono _ \l_tmpa_tl _ tl
I204     },
I205     ipa~ mono~ #1~ features
I206     .code:n              = {
I207       \tl_set:Nn \l_tmpb_tl { #1 }
I208       \tl_replace_all:Nnn \l_tmpb_tl { ~ } { _ }
I209       \tl_put_right:ce {
I210         g _ lngx _ ipa _ mono _ \l_tmpb_tl _ features _ tl
I211       } { ##1 , }
I212       \tl_clear:N \l_tmpb_tl
I213     }
I214   }
I215   \tl_clear:N \l_tmpa_tl
I216 }

```

(End of definition for *ipa sans upright* and others. These functions are documented on page 8.)

These variables store the values for fonts and features for the monospaced IPA.

```

ipa mono upright
ipa mono upright features
ipa mono bold upright
ipa mono bold upright features
ipa mono italic
ipa mono italic features
ipa mono bold italic
ipa mono bold italic features
ipa mono slanted
ipa mono slanted features
ipa mono bold slanted
ipa mono bold slanted features
ipa mono swash
ipa mono swash features
ipa mono bold swash
ipa mono bold swash features
ipa mono small caps
ipa mono small caps features
\g_lngx_ipa_mono_upright_tl
\g_lngx_ipa_mono_upright_features_tl
\g_lngx_ipa_mono_bold_upright_tl
\g_lngx_ipa_mono_bold_upright_features_tl
\g_lngx_ipa_mono_italic_tl
\g_lngx_ipa_mono_italic_features_tl
\g_lngx_ipa_mono_bold_italic_tl
\g_lngx_ipa_mono_bold_italic_features_tl
\g_lngx_ipa_mono_slanted_tl
\g_lngx_ipa_mono_slanted_features_tl
\g_lngx_ipa_mono_bold_slanted_tl
\g_lngx_ipa_mono_bold_slanted_features_tl
\g_lngx_ipa_mono_swash_tl
\g_lngx_ipa_mono_swash_features_tl
\g_lngx_ipa_mono_bold_swash_tl
\g_lngx_ipa_mono_bold_swash_features_tl
\g_lngx_ipa_mono_small_caps_tl
\g_lngx_ipa_mono_small_caps_features_tl
ipa newcm
ipa newcm features
ipa newcm bold
ipa newcm bold features
ipa newcm italic
ipa newcm italic features

```

(End of definition for *ipa mono upright* and others. These functions are documented on page 8.)

This key sets New Computer Modern fonts in all weights, all families in the context of IPA.

```

\keys_define:nn { lngx _ keys } {
  ipa~ newcm
  .meta:n = {
    ipa~
    upright = {
      NewCM10-Book.otf
    },
    ipa~
    bold~ upright = {
      NewCM10-Bold.otf
    },
    ipa~
    italic = {

```

```

1265     NewCM10-BookItalic.otf
1266 },
1267 ipa~
1268 bold~ italic          = {
1269     NewCM10-BoldItalic.otf
1270 },
1271 ipa~
1272 slanted                = {
1273     NewCM10-Book.otf
1274 },
1275 ipa~
1276 bold~ slanted          = {
1277     NewCM10-Bold.otf
1278 },
1279 ipa~
1280 swash                  = {
1281     NewCM10-Book.otf
1282 },
1283 ipa~
1284 bold~ swash            = {
1285     NewCM10-Bold.otf
1286 },
1287 ipa~
1288 small~ caps            = {
1289     NewCM10-Book.otf
1290 },
1291 ipa~
1292 sans~ upright          = {
1293     NewCMSans10-Book.otf
1294 },
1295 ipa~
1296 sans~ bold~ upright    = {
1297     NewCMSans10-Bold.otf
1298 },
1299 ipa~
1300 sans~ italic           = {
1301     NewCMSans10-BookOblique.otf
1302 },
1303 ipa~
1304 sans~ bold~ italic     = {
1305     NewCMSans10-BoldOblique.otf
1306 },
1307 ipa~
1308 sans~ slanted          = {
1309     NewCMSans10-BookOblique.otf
1310 },
1311 ipa~
1312 sans~ bold~ slanted    = {
1313     NewCMSans10-BoldOblique.otf
1314 },
1315 ipa~
1316 sans~ swash            = {
1317     NewCMSans10-Book.otf
1318 },

```



```

1319 ipa~
1320 sans~ bold~ swash      = {
1321     NewCMSans10-Bold.otf
1322 },
1323 ipa~
1324 sans~ small~ caps      = {
1325     NewCMSans10-Book.otf
1326 },
1327 ipa~
1328 mono~ upright          = {
1329     NewCMMono10-Book.otf
1330 },
1331 ipa~
1332 mono~ bold~ upright    = {
1333     NewCMMono10-Bold.otf
1334 },
1335 ipa~
1336 mono~ italic           = {
1337     NewCMMono10-BookItalic.otf
1338 },
1339 ipa~
1340 mono~ bold~ italic     = {
1341     NewCMMono10-BoldOblique.otf
1342 },
1343 ipa~
1344 mono~ slanted           = {
1345     NewCMMono10-Book.otf
1346 },
1347 ipa~
1348 mono~ bold~ slanted    = {
1349     NewCMMono10-BoldOblique.otf
1350 },
1351 ipa~
1352 mono~ swash            = {
1353     NewCMMono10-Book.otf
1354 },
1355 ipa~
1356 mono~ bold~ swash      = {
1357     NewCMMono10-Bold.otf
1358 },
1359 ipa~
1360 mono~ small~ caps      = {
1361     NewCMMono10-Book.otf
1362 }
1363 }
1364 }

```

(End of definition for *ipa newcm*. This function is documented on page 7.)

ipa newcm sans This key sets New Computer Modern sans fonts in all weights, all families in the context of IPA.

```

1365 \keys_define:nn { lngx _ keys } {
1366   ipa~ newcm~ sans
1367 }

```

```

1368 .meta:n          = {
1369   ipa~
1370   upright          = {
1371     NewCMSans10-Book.otf
1372   },
1373   ipa~
1374   bold~ upright    = {
1375     NewCMSans10-Bold.otf
1376   },
1377   ipa~
1378   italic           = {
1379     NewCMSans10-BookOblique.otf
1380   },
1381   ipa~
1382   bold~ italic     = {
1383     NewCMSans10-BoldOblique.otf
1384   },
1385   ipa~
1386   slanted          = {
1387     NewCMSans10-BookOblique.otf
1388   },
1389   ipa~
1390   bold~ slanted    = {
1391     NewCMSans10-BoldOblique.otf
1392   },
1393   ipa~
1394   swash            = {
1395     NewCMSans10-Book.otf
1396   },
1397   ipa~
1398   bold~ swash      = {
1399     NewCMSans10-Bold.otf
1400   },
1401   ipa~
1402   small~ caps      = {
1403     NewCMSans10-Book.otf
1404   }
1405 }
1406 }

```

(End of definition for *ipa newcm sans*. This function is documented on page 7.)

ipa newcm mono This key sets New Computer Modern monospaced fonts in all weights, all families in the context of IPA.

```

1407 \keys_define:nn { lngx _ keys } {
1408   ipa~ newcm~ mono
1409   .meta:n          = {
1410     ipa~
1411     upright          = {
1412       NewCMMono10-Book.otf
1413     },
1414     ipa~
1415     bold~ upright    = {
1416

```

```

I417     NewCMMono10-Bold.otf
I418 },
I419 ipa~
I420 italic                = {
I421     NewCMMono10-BookItalic.otf
I422 },
I423 ipa~
I424 bold~ italic          = {
I425     NewCMMono10-BoldOblique.otf
I426 },
I427 ipa~
I428 slanted                = {
I429     NewCMMono10-Book.otf
I430 },
I431 ipa~
I432 bold~ slanted          = {
I433     NewCMMono10-BoldOblique.otf
I434 },
I435 ipa~
I436 swash                  = {
I437     NewCMMono10-Book.otf
I438 },
I439 ipa~
I440 bold~ swash            = {
I441     NewCMMono10-Bold.otf
I442 },
I443 ipa~
I444 small~ caps            = {
I445     NewCMMono10-Book.otf
I446 }
I447 }
I448 }

```

(End of definition for *ipa newcm mono*. This function is documented on page 7.)

ipa newcm regular This key sets New Computer Modern regular serif fonts in all weights, all families in the context of IPA.

```

I449
I450 \keys_define:nn { lngx _ keys } {
I451   ipa~ newcm~ regular
I452   .meta:n                = {
I453     ipa~
I454     upright                = {
I455         NewCM10-Regular.otf
I456     },
I457     ipa~
I458     bold~ upright          = {
I459         NewCM10-Bold.otf
I460     },
I461     ipa~
I462     italic                = {
I463         NewCM10-Italic.otf
I464     },
I465     ipa~

```

```

1466 bold~ italic          = {
1467     NewCM10-BoldItalic.otf
1468 },
1469 ipa~
1470 slanted                = {
1471     NewCM10-Regular.otf
1472 },
1473 ipa~
1474 bold~ slanted          = {
1475     NewCM10-Bold.otf
1476 },
1477 ipa~
1478 swash                  = {
1479     NewCM10-Regular.otf
1480 },
1481 ipa~
1482 bold~ swash            = {
1483     NewCM10-Bold.otf
1484 },
1485 ipa~
1486 small~ caps            = {
1487     NewCM10-Regular.otf
1488 },
1489 ipa~
1490 sans~ upright          = {
1491     NewCMSans10-Regular.otf
1492 },
1493 ipa~
1494 sans~ bold             = {
1495     NewCMSans10-Bold.otf
1496 },
1497 ipa~
1498 sans~ italic           = {
1499     NewCMSans10-Oblique.otf
1500 },
1501 ipa~
1502 sans~ bold~ italic     = {
1503     NewCMSans10-BoldOblique.otf
1504 },
1505 ipa~
1506 sans~ slanted          = {
1507     NewCMSans10-Regular.otf
1508 },
1509 ipa~
1510 sans~ bold~ slanted    = {
1511     NewCMSans10-Bold.otf
1512 },
1513 ipa~
1514 sans~ swash            = {
1515     NewCMSans10-Regular.otf
1516 },
1517 ipa~
1518 sans~ bold~ swash      = {
1519     NewCMSans10-Bold.otf

```

```

1520 },
1521 ipa~
1522 sans~ small~ caps      = {
1523     NewCMSans10-Regular.otf
1524 },
1525 ipa~
1526 mono~ upright          = {
1527     NewCMMono10-Regular.otf
1528 },
1529 ipa~
1530 mono~ bold             = {
1531     NewCMMono10-Bold.otf
1532 },
1533 ipa~
1534 mono~ italic           = {
1535     NewCMMono10-Italic.otf
1536 },
1537 ipa~
1538 mono~ bold~ italic     = {
1539     NewCMMono10-Bold.otf
1540 },
1541 ipa~
1542 mono~ slanted          = {
1543     NewCMMono10-Regular.otf
1544 },
1545 ipa~
1546 mono~ bold~ slanted    = {
1547     NewCMMono10-Bold.otf
1548 },
1549 ipa~
1550 mono~ swash            = {
1551     NewCMMono10-Regular.otf
1552 },
1553 ipa~
1554 mono~ bold~ swash      = {
1555     NewCMMono10-Bold.otf
1556 },
1557 ipa~
1558 mono~ small~ caps      = {
1559     NewCMMono10-Regular.otf
1560 }
1561 }
1562 }

```

(End of definition for *ipa newcm regular*. This function is documented on page 7.)

ipa newcm regular sans This key sets New Computer Modern regular sans fonts in all weights, all families in the context of IPA.

```

1563
1564 \keys_define:nn { lngx _ keys } {
1565     ipa~ newcm~ sans~ regular
1566     .meta:n          = {
1567         ipa~
1568         upright      = {

```

```

1569     NewCMSans10-Regular.otf
1570 },
1571 ipa~
1572 bold = {
1573     NewCMSans10-Bold.otf
1574 },
1575 ipa~
1576 italic = {
1577     NewCMSans10-Oblique.otf
1578 },
1579 ipa~
1580 bold~ italic = {
1581     NewCMSans10-BoldOblique.otf
1582 },
1583 ipa~
1584 slanted = {
1585     NewCMSans10-Regular.otf
1586 },
1587 ipa~
1588 bold~ slanted = {
1589     NewCMSans10-Bold.otf
1590 },
1591 ipa~
1592 swash = {
1593     NewCMSans10-Regular.otf
1594 },
1595 ipa~
1596 bold~ swash = {
1597     NewCMSans10-Bold.otf
1598 },
1599 ipa~
1600 small~ caps = {
1601     NewCMSans10-Regular.otf
1602 }
1603 }
1604 }

```

(End of definition for *ipa newcm regular sans*. This function is documented on page 7.)

ipa newcm regular mono This key sets New Computer Modern regular monospaced fonts in all weights, all families in the context of IPA.

```

1605
1606 \keys_define:nn { lngx _ keys } {
1607   ipa~ newcm~ mono~ regular
1608   .meta:n = {
1609     ipa~
1610     upright = {
1611         NewCMMono10-Regular.otf
1612     },
1613     ipa~
1614     bold = {
1615         NewCMMono10-Bold.otf
1616     },
1617     ipa~

```

```

1618     italic                = {
1619       NewCMMono10-Italic.otf
1620     },
1621     ipa~
1622     bold~ italic          = {
1623       NewCMMono10-Bold.otf
1624     },
1625     ipa~
1626     slanted               = {
1627       NewCMMono10-Regular.otf
1628     },
1629     ipa~
1630     bold~ slanted         = {
1631       NewCMMono10-Bold.otf
1632     },
1633     ipa~
1634     swash                 = {
1635       NewCMMono10-Regular.otf
1636     },
1637     ipa~
1638     bold~ swash           = {
1639       NewCMMono10-Bold.otf
1640     },
1641     ipa~
1642     small~ caps           = {
1643       NewCMMono10-Regular.otf
1644     }
1645   }
1646 }

```

(End of definition for *ipa newcm regular mono*. This function is documented on page 7.)

We set the `ipa newcm` key by default.

```

1647
1648 \ltx_set_keys:n {ipa~ newcm}

```

If Lua^AT_EX is loaded, the `HarfBuzz` renderer is selected by default.

```

1649
1650 \sys_if_engine luatex:T {
1651   \ltx_set_keys:n {
1652     ipa~
1653     upright~ features    = {
1654       Renderer           = { HarfBuzz }
1655     },
1656     ipa~ sans~
1657     upright~ features    = {
1658       Renderer           = { HarfBuzz }
1659     },
1660     ipa~ mono~
1661     upright~ features    = {
1662       Renderer           = { HarfBuzz }
1663     }
1664   }
1665 }

```

`\lngx_set_main_ipa_font:nn` Here, I develop font-setting commands for IPA. These commands are set with
`\lngx_main_ipa:` `\setfontfamily`, so they keep overriding the definitions of the same command names.
`lngx_ipa_rm_nfss` These commands set NFSS families that we use later for setting the IPA fonts. These
`\lngx_set_sans_ipa_font:nn` functions and NFSS families are public, but manipulating them has effects (mostly desired)
`\lngx_sans_ipa:` at several other places, so use them with caution.
`lngx_ipa_sf_nfss`

```

1666
1667 \cs_new_protected:Npn \lngx_set_main_ipa_font:nn #1#2 {
1668   \setfontfamily \lngx_main_ipa: [
1669     #1,
1670     NFSSFamily           = { lngx _ ipa _ rm _ nfss }
1671   ] { #2 }
1672 }
1673
1674 \cs_new_protected:Npn \lngx_set_sans_ipa_font:nn #1#2 {
1675   \setfontfamily \lngx_sans_ipa: [
1676     #1,
1677     NFSSFamily           = { lngx _ ipa _ sf _ nfss }
1678   ] { #2 }
1679 }
1680
1681 \cs_new_protected:Npn \lngx_set_mono_ipa_font:nn #1#2 {
1682   \setfontfamily \lngx_mono_ipa: [
1683     #1,
1684     NFSSFamily           = { lngx _ ipa _ tt _ nfss }
1685   ] { #2 }
1686 }
1687
1688 \cs_generate_variant:Nn \lngx_set_main_ipa_font:nn { ee }
1689 \cs_generate_variant:Nn \lngx_set_sans_ipa_font:nn { ee }
1690 \cs_generate_variant:Nn \lngx_set_mono_ipa_font:nn { ee }

```

(End of definition for `\lngx_set_main_ipa_font:nn` and others. These functions are documented on page 13.)

`lngx_ipa` Here, I create a ‘super font family’ with `\lngx_super_font_family:nn`, a macro provided by LINGUIS~~TL~~X-NFSS. Please see the documentation of that package for more information. Note that `lngx_ipa` is a super family responsible for all the IPA-related functions of the package. It is associated with the NFSS families defined just now for the IPA.

```

1691
1692 \lngx_super_font_family:nn { lngx _ ipa } {
1693   rm           = { lngx _ ipa _ rm _ nfss },
1694   sf           = { lngx _ ipa _ sf _ nfss },
1695   tt           = { lngx _ ipa _ tt _ nfss }
1696 }

```

(End of definition for `lngx_ipa`. This function is documented on page 13.)

`\lngxipa` I use `\lngx softer super font family:n` provided by LINGUIS~~TL~~X-NFSS for defining this
`\lngx_ipa:` switch to the IPA.

```

1697
1698 \cs_new_protected:Npn \lngx_ipa: {
1699   \lngx softer super font family:n { lngx _ ipa }
1700 }
1701
1702 \cs_set_eq:NN \lngxipa \lngx_ipa:

```


(End of definition for `\lngxipa` and `\lngx_ipa:`. These functions are documented on page 7.)

Now, I have used the exact same method that I described in the implementation of `LINGUISC-X-FONTS` for setting the size variants. This is done with lazy evaluation, just before `\begin{document}`.

```

1703
1704 \hook_gput_code:nnn { begindocument / before } { . } {
1705   \tl_if_in:cnT {
1706     g _ lngx _ ipa _ upright _ tl
1707   } { NewCM } {
1708     \tl_if_in:cnTF {
1709       g _ lngx _ ipa _ upright _ tl
1710     } { Book } {
1711       \lngx_set_keys:n {
1712         ipa~
1713         upright~ features      = {
1714           SizeFeatures        = {
1715             {
1716               Size              = {-8},
1717               Font              = {
1718                 NewCM08-Book.otf
1719             }
1720           },
1721           {
1722             Size                = {8-},
1723             Font                = {
1724               NewCM10-Book.otf
1725             }
1726           }
1727         }
1728       }
1729     } {
1730       \lngx_set_keys:n {
1731         ipa~
1732         upright~ features      = {
1733           SizeFeatures        = {
1734             {
1735               Size              = {-8},
1736               Font              = {
1737                 NewCM08-Regular.otf
1738             }
1739           },
1740           {
1741             Size                = {8-},
1742             Font                = {
1743               NewCM10-Regular.otf
1744             }
1745           }
1746         }
1747       }
1748     }
1749   }
1750 }
1751 \tl_if_in:cnT {
1752
```

```

1753     g _ lngx _ ipa _ upright _ tl
1754 } { NewCMSans } {
1755     \tl_if_in:cnTF {
1756         g _ lngx _ ipa _ upright _ tl
1757     } { Book } {
1758         \lngx_set_keys:n {
1759             ipa~
1760             upright~ features      = {
1761                 SizeFeatures      = {
1762                     {
1763                         Size          = {-8},
1764                         Font          = {
1765                             NewCMSans08-Book.otf
1766                         }
1767                     },
1768                     {
1769                         Size          = {8-},
1770                         Font          = {
1771                             NewCMSans10-Book.otf
1772                         }
1773                     }
1774                 }
1775             }
1776         }
1777     } {
1778         \lngx_set_keys:n {
1779             ipa~
1780             upright~ features      = {
1781                 SizeFeatures      = {
1782                     {
1783                         Size          = {-8},
1784                         Font          = {
1785                             NewCMSans08-Regular.otf
1786                         }
1787                     },
1788                     {
1789                         Size          = {8-},
1790                         Font          = {
1791                             NewCMSans10-Regular.otf
1792                         }
1793                     }
1794                 }
1795             }
1796         }
1797     }
1798 }
1799 \tl_if_in:cnT {
1800     g _ lngx _ ipa _ italic _ tl
1801 } { NewCM } {
1802     \tl_if_in:cnTF {
1803         g _ lngx _ ipa _ italic _ tl
1804     } { Book } {
1805         \lngx_set_keys:n {
1806             ipa~

```

```

1807         italic~ features      = {
1808             SizeFeatures      = {
1809                 {
1810                     Size        = {-8},
1811                     Font        = {
1812                         NewCM08-BookItalic.otf
1813                     }
1814                 },
1815                 {
1816                     Size        = {8-},
1817                     Font        = {
1818                         NewCM10-BookItalic.otf
1819                     }
1820                 }
1821             }
1822         }
1823     }
1824 } {
1825     \lngx_set_keys:n {
1826         ipa~
1827         italic~ features      = {
1828             SizeFeatures      = {
1829                 {
1830                     Size        = {-8},
1831                     Font        = {
1832                         NewCM08-Italic.otf
1833                     }
1834                 },
1835                 {
1836                     Size        = {8-},
1837                     Font        = {
1838                         NewCM10-Italic.otf
1839                     }
1840                 }
1841             }
1842         }
1843     }
1844 }
1845 }
1846 \tl_if_in:cnT {
1847     g _ lngx _ ipa _ italic _ tl
1848 } { NewCMSans } {
1849     \tl_if_in:cnTF {
1850         g _ lngx _ ipa _ italic _ tl
1851     } { Book } {
1852         \lngx_set_keys:n {
1853             ipa~
1854             italic~ features      = {
1855                 SizeFeatures      = {
1856                     {
1857                         Size        = {-8},
1858                         Font        = {
1859                             NewCMSans08-BookOblique.otf
1860                     }

```

```

1861         },
1862         {
1863             Size                = {8-},
1864             Font                 = {
1865                 NewCMSans10-BookOblique.otf
1866             }
1867         }
1868     }
1869 }
1870 }
1871 } {
1872     \lngx_set_keys:n {
1873         ipa~
1874         italic~ features      = {
1875             SizeFeatures      = {
1876                 {
1877                     Size                = {-8},
1878                     Font                 = {
1879                         NewCMSans08-Oblique.otf
1880                     }
1881                 },
1882                 {
1883                     Size                = {8-},
1884                     Font                 = {
1885                         NewCMSans10-Oblique.otf
1886                     }
1887                 }
1888             }
1889         }
1890     }
1891 }
1892 }
1893 \tl_if_in:cnT {
1894     g _ lngx _ ipa _ sans _ upright _ tl
1895 } { NewCMSans } {
1896     \tl_if_in:cnTF {
1897         g _ lngx _ ipa _ upright _ tl
1898     } { Book } {
1899         \lngx_set_keys:n {
1900             ipa~ sans~
1901             upright~ features      = {
1902                 SizeFeatures      = {
1903                     {
1904                         Size                = {-8},
1905                         Font                 = {
1906                             NewCMSans08-Book.otf
1907                         }
1908                     },
1909                     {
1910                         Size                = {8-},
1911                         Font                 = {
1912                             NewCMSans10-Book.otf
1913                         }
1914                     }
1915                 }
1916             }
1917         }
1918     }
1919 }

```

```

1915     }
1916   }
1917 }
1918 } {
1919   \lngx_set_keys:n {
1920     ipa~ sans~
1921     upright~ features      = {
1922       SizeFeatures        = {
1923         {
1924           Size             = {-8},
1925           Font             = {
1926             NewCMSans08-Regular.otf
1927           }
1928         },
1929         {
1930           Size             = {8-},
1931           Font             = {
1932             NewCMSans10-Regular.otf
1933           }
1934         }
1935       }
1936     }
1937   }
1938 }
1939 }
1940 \tl_if_in:cnT {
1941   g _ lngx _ ipa _ sans _ italic _ tl
1942 } { NewCMSans } {
1943   \tl_if_in:cnTF {
1944     g _ lngx _ ipa _ italic _ tl
1945   } { Book } {
1946     \lngx_set_keys:n {
1947       ipa~ sans~
1948       italic~ features    = {
1949         SizeFeatures      = {
1950           {
1951             Size           = {-8},
1952             Font           = {
1953               NewCMSans08-BookOblique.otf
1954             }
1955           },
1956           {
1957             Size           = {8-},
1958             Font           = {
1959               NewCMSans10-BookOblique.otf
1960             }
1961           }
1962         }
1963       }
1964     }
1965   } {
1966     \lngx_set_keys:n {
1967       ipa~ sans~
1968       italic~ features    = {

```

```

1969         SizeFeatures          = {
1970             {
1971                 Size              = {-8},
1972                 Font              = {
1973                     NewCMSans08-Oblique.otf
1974                 }
1975             },
1976             {
1977                 Size              = {8-},
1978                 Font              = {
1979                     NewCMSans10-Oblique.otf
1980                 }
1981             }
1982         }
1983     }
1984 }
1985 }
1986 }

```

Now, I set the keys with the appropriate values and end the package.

```

1987 \lngx_set_keys:n {
1988     ipa~ upright~ features = {
1989         UprightFont        = {
1990             \g_lngx_ipa_upright_tl
1991         },
1992         UprightFeatures    = {
1993             \g_lngx_ipa_upright_features_tl
1994         },
1995         BoldFont           = {
1996             \g_lngx_ipa_bold_upright_tl
1997         },
1998         BoldFeatures       = {
1999             \g_lngx_ipa_bold_upright_features_tl
2000         },
2001         ItalicFont         = {
2002             \g_lngx_ipa_italic_tl
2003         },
2004         ItalicFeatures     = {
2005             \g_lngx_ipa_italic_features_tl
2006         },
2007         BoldItalicFont     = {
2008             \g_lngx_ipa_bold_italic_tl
2009         },
2010         BoldItalicFeatures = {
2011             \g_lngx_ipa_bold_italic_features_tl
2012         },
2013     \tl_if_empty:cF {
2014         g _ lngx _ ipa _ slanted _ tl
2015     } {
2016         SlantedFont        = {
2017             \g_lngx_ipa_slanted_tl
2018         },
2019     \tl_if_empty:cF {
2020         g _ lngx _ ipa _ slanted _ features _ tl
2021     } {

```

```

2022         SlantedFeatures           = {
2023             \g_lngx_ipa_slanted_features_tl
2024         },
2025     },
2026 }
2027 \tl_if_empty:cF {
2028     g _ lngx _ ipa _ bold _ slanted _ tl
2029 } {
2030     BoldSlantedFont           = {
2031         \g_lngx_ipa_bold_slanted_tl
2032     },
2033     \tl_if_empty:cF {
2034         g _ lngx _ ipa _ bold _ slanted _ features _ tl
2035     } {
2036         BoldSlantedFeatures    = {
2037             \g_lngx_ipa_bold_slanted_features_tl
2038         },
2039     }
2040 }
2041 \tl_if_empty:cF {
2042     g _ lngx _ ipa _ swash _ tl
2043 } {
2044     SwashFont                 = {
2045         \g_lngx_ipa_swash_tl
2046     },
2047     \tl_if_empty:cF {
2048         g _ lngx _ ipa _ swash _ features _ tl
2049     } {
2050         SwashFeatures          = {
2051             \g_lngx_ipa_swash_features_tl
2052         },
2053     }
2054 }
2055 \tl_if_empty:cF {
2056     g _ lngx _ ipa _ bold _ swash _ tl
2057 } {
2058     BoldSwashFont             = {
2059         \g_lngx_ipa_bold_swash_tl
2060     },
2061     \tl_if_empty:cF {
2062         g _ lngx _ ipa _ bold _ swash _ features _ tl
2063     } {
2064         BoldSwashFeatures      = {
2065             \g_lngx_ipa_bold_swash_features_tl
2066         },
2067     }
2068 }
2069 \tl_if_empty:cF {
2070     g _ lngx _ ipa _ small _ caps _ tl
2071 } {
2072     SmallCapsFont             = {
2073         \g_lngx_ipa_small_caps_tl
2074     }
2075     \tl_if_empty:cF {

```

```

2076         g _ lngx _ ipa _ small _ caps _ features _ tl
2077     } {
2078         SmallCapsFeatures          = {
2079             \g_lngx_ipa_small_caps_features_tl
2080         }
2081     }
2082 }
2083 },
2084 ipa~
2085 sans~ upright~ features = {
2086     UprightFont                  = {
2087         \g_lngx_ipa_sans_upright_tl
2088     },
2089     UprightFeatures              = {
2090         \g_lngx_ipa_sans_upright_features_tl
2091     },
2092     BoldFont                     = {
2093         \g_lngx_ipa_sans_bold_upright_tl
2094     },
2095     BoldFeatures                 = {
2096         \g_lngx_ipa_sans_bold_upright_features_tl
2097     },
2098     ItalicFont                   = {
2099         \g_lngx_ipa_sans_italic_tl
2100     },
2101     ItalicFeatures               = {
2102         \g_lngx_ipa_sans_italic_features_tl
2103     },
2104     BoldItalicFont              = {
2105         \g_lngx_ipa_sans_bold_italic_tl
2106     },
2107     BoldItalicFeatures          = {
2108         \g_lngx_ipa_sans_bold_italic_features_tl
2109     },
2110     \tl_if_empty:cF {
2111         g _ lngx _ ipa _ slanted _ tl
2112     } {
2113         SlantedFont              = {
2114             \g_lngx_ipa_slanted_tl
2115         },
2116         \tl_if_empty:cF {
2117             g _ lngx _ ipa _ slanted _ features _ tl
2118         } {
2119             SlantedFeatures      = {
2120                 \g_lngx_ipa_slanted_features_tl
2121             },
2122         }
2123     }
2124     \tl_if_empty:cF {
2125         g _ lngx _ ipa _ sans _ bold _ slanted _ tl
2126     } {
2127         BoldSlantedFont          = {
2128             \g_lngx_ipa_sans_bold_slanted_tl
2129         },

```



```

2130     \tl_if_empty:cF {
2131         g _ lngx _ ipa _ sans _ bold _ slanted _ features
2132         _ tl
2133     } {
2134         BoldSlantedFeatures = {
2135             \g_lngx_ipa_sans_bold_slanted_features_tl
2136         },
2137     }
2138 }
2139 \tl_if_empty:cF {
2140     g _ lngx _ ipa _ sans _ swash _ tl
2141 } {
2142     SwashFont = {
2143         \g_lngx_ipa_sans_swash_tl
2144     },
2145     \tl_if_empty:cF {
2146         g _ lngx _ ipa _ sans _ swash _ features _ tl
2147     } {
2148         SwashFeatures = {
2149             \g_lngx_ipa_sans_swash_features_tl
2150         },
2151     }
2152 }
2153 \tl_if_empty:cF {
2154     g _ lngx _ ipa _ sans _ bold _ swash _ tl
2155 } {
2156     BoldSwashFont = {
2157         \g_lngx_ipa_sans_bold_swash_tl
2158     },
2159     \tl_if_empty:cF {
2160         g _ lngx _ ipa _ sans _ bold _ swash _ features _
2161         tl
2162     } {
2163         BoldSwashFeatures = {
2164             \g_lngx_ipa_sans_bold_swash_features_tl
2165         },
2166     }
2167 }
2168 \tl_if_empty:cF {
2169     g _ lngx _ ipa _ sans _ small _ caps _ tl
2170 } {
2171     SmallCapsFont = {
2172         \g_lngx_ipa_sans_small_caps_tl
2173     }
2174     \tl_if_empty:cF {
2175         g _ lngx _ ipa _ sans _ small _ caps _ features _
2176         tl
2177     } {
2178         SmallCapsFeatures = {
2179             \g_lngx_ipa_sans_small_caps_features_tl
2180         }
2181     }
2182 }
2183 },

```

```

2184 ipa~
2185 mono~ upright~ features = {
2186     UprightFont = {
2187         \g_lngx_ipa_mono_upright_tl
2188     },
2189     UprightFeatures = {
2190         \g_lngx_ipa_mono_upright_features_tl
2191     },
2192     BoldFont = {
2193         \g_lngx_ipa_mono_bold_upright_tl
2194     },
2195     BoldFeatures = {
2196         \g_lngx_ipa_mono_bold_upright_features_tl
2197     },
2198     ItalicFont = {
2199         \g_lngx_ipa_mono_italic_tl
2200     },
2201     ItalicFeatures = {
2202         \g_lngx_ipa_mono_italic_features_tl
2203     },
2204     BoldItalicFont = {
2205         \g_lngx_ipa_mono_bold_italic_tl
2206     },
2207     BoldItalicFeatures = {
2208         \g_lngx_ipa_mono_bold_italic_features_tl
2209     },
2210     \tl_if_empty:cF {
2211         g _ lngx _ ipa _ mono _ slanted _ tl
2212     } {
2213         SlantedFont = {
2214             \g_lngx_ipa_mono_slanted_tl
2215         },
2216         \tl_if_empty:cF {
2217             g _ lngx _ ipa _ mono _ slanted _ features _ tl
2218         } {
2219             SlantedFeatures = {
2220                 \g_lngx_ipa_mono_slanted_features_tl
2221             },
2222         }
2223     }
2224     \tl_if_empty:cF {
2225         g _ lngx _ ipa _ mono _ bold _ slanted _ tl
2226     } {
2227         BoldSlantedFont = {
2228             \g_lngx_ipa_mono_bold_slanted_tl
2229         },
2230         \tl_if_empty:cF {
2231             g _ lngx _ ipa _ mono _ bold _ slanted _ features
2232             _ tl
2233         } {
2234             BoldSlantedFeatures = {
2235                 \g_lngx_ipa_mono_bold_slanted_features_tl
2236             },
2237         }

```

```

2238 }
2239 \tl_if_empty:cF {
2240   g _ lngx _ ipa _ mono _ swash _ tl
2241 } {
2242   SwashFont = {
2243     \g_lngx_ipa_mono_swash_tl
2244   },
2245   \tl_if_empty:cF {
2246     g _ lngx _ ipa _ mono _ swash _ features _ tl
2247   } {
2248     SwashFeatures = {
2249       \g_lngx_ipa_mono_swash_features_tl
2250     },
2251   }
2252 }
2253 \tl_if_empty:cF {
2254   g _ lngx _ ipa _ mono _ bold _ swash _ tl
2255 } {
2256   BoldSwashFont = {
2257     \g_lngx_ipa_mono_bold_swash_tl
2258   },
2259   \tl_if_empty:cF {
2260     g _ lngx _ ipa _ mono _ bold _ swash _ features _
2261     tl
2262   } {
2263     BoldSwashFeatures = {
2264       \g_lngx_ipa_mono_bold_swash_features_tl
2265     },
2266   }
2267 }
2268 \tl_if_empty:cF {
2269   g _ lngx _ ipa _ mono _ small _ caps _ tl
2270 } {
2271   SmallCapsFont = {
2272     \g_lngx_ipa_mono_small_caps_tl
2273   }
2274   \tl_if_empty:cF {
2275     g _ lngx _ ipa _ mono _ small _ caps _ features _
2276     tl
2277   } {
2278     SmallCapsFeatures = {
2279       \g_lngx_ipa_mono_small_caps_features_tl
2280     }
2281   }
2282 }
2283 }
2284 }
2285 \tl_if_in:cnT {
2286   g _ lngx _ ipa _ upright _ tl
2287 } { NewCM } {
2288   \lngx_set_keys:n {
2289     ipa~
2290     upright~ features = {
2291       IgnoreFontspecFile,

```

```

2292         StylisticSet                = { 5 }
2293     }
2294 }
2295 }
2296 \tl_if_in:cnT {
2297     g _ lngx _ ipa _ sans _ upright _ tl
2298 } { NewCM } {
2299     \lngx_set_keys:n {
2300         ipa~ sans~
2301         upright~ features                = {
2302             IgnoreFontspecFile,
2303             StylisticSet                = { 5 }
2304         }
2305     }
2306 }
2307 \tl_if_in:cnT {
2308     g _ lngx _ ipa _ mono _ upright _ tl
2309 } { NewCM } {
2310     \lngx_set_keys:n {
2311         ipa~ mono~
2312         upright~ features                = {
2313             IgnoreFontspecFile,
2314             StylisticSet                = { 5 }
2315         }
2316     }
2317 }
2318 \lngx_set_main_ipa_font:ee {
2319     \g_lngx_ipa_upright_features_tl
2320 } {
2321     \g_lngx_ipa_upright_tl
2322 }
2323 \lngx_set_sans_ipa_font:ee {
2324     \g_lngx_ipa_sans_upright_features_tl
2325 } {
2326     \g_lngx_ipa_sans_upright_tl
2327 }
2328 \lngx_set_mono_ipa_font:ee {
2329     \g_lngx_ipa_mono_upright_features_tl
2330 } {
2331     \g_lngx_ipa_mono_upright_tl
2332 }
2333 }
2334 </ipa>

```

```

2335 <*logos>
2336 \ProvidesExplPackage{linguistix-logos}
2337     {2025-06-19}
2338     {v0.5a}
2339     {%
2340         Logos of the ‘LinguisTiX’ bundle..%
2341     }

```

The fontspec package (if not already loaded).

```

2342
2343 \IfPackageLoadedF { fontspec } {
2344     \RequirePackage { fontspec }
2345 }

```

\lngx_logo_font: This is a command that switches to the New Computer Modern Uncial font family.

```

2346
2347 \newfontfamily \lngx_logo_font: [
2348     IgnoreFontspecFile,
2349     UprightFont          = { NewCMUncial10-Book.otf },
2350     UprightFeatures      = {
2351         SizeFeatures      = {
2352             {
2353                 Size          = {-8},
2354                 Font          = {NewCMUncial08-Book.otf}
2355             },
2356             {
2357                 Size          = {8-},
2358                 Font          = {NewCMUncial10-Book.otf}
2359             },
2360         }
2361     },
2362     BoldFont              = { NewCMUncial10-Bold.otf },
2363     BoldFeatures          = {
2364         SizeFeatures      = {
2365             {
2366                 Size          = {-8},
2367                 Font          = {NewCMUncial08-Bold.otf}
2368             },
2369             {
2370                 Size          = {8-},
2371                 Font          = {NewCMUncial10-Bold.otf}
2372             },
2373         }
2374     },
2375     Renderer              = { HarfBuzz }
2376 ]{ NewCMUncial10-Book.otf }

```

(End of definition for \lngx_logo_font:. This function is documented on page 15.)

lngx_purple_color

```

2377
2378 \color_set:nn { lngx _ purple _ color } { blue ! 50 ! red }

```

(End of definition for lngx_purple_color. This function is documented on page 15.)

\lngxlogo

```
2379
2380 \NewDocumentCommand \lngxlogo { 0{ } } {%
2381   \group_begin:
2382   \lngx_logo_font:
2383   LinguisTi
2384   \color_group_begin:
2385   \color_select:n { lngx_purple_color }
2386   X
2387   \color_group_end:
2388   \IfBlankF { #1 } { - #1 }
2389   \group_end:
2390 }
```

(End of definition for \lngxlogo. This function is documented on page 9.)

\lngxpkg

```
2391
2392 \cs_new:Npn \lngxpkg {
2393   \IfPackageLoadedTF { hyperref } {
2394     \texorpdfstring {
2395       \lngxlogo
2396     } {
2397       LinguisTiX
2398     }
2399   } {
2400     \lngxlogo
2401   }
2402 }
```

(End of definition for \lngxpkg. This function is documented on page 9.)

\lngxbaselogo

\lngxfontslogo

\lngxipalogo

\lngxlogoslogo

\lngxnfsslogo

```
2403
2404 \clist_map_inline:nn {
2405   base,
2406   examples,
2407   fixpex,
2408   fonts,
2409   ipa,
2410   logos,
2411   nfss
2412 } {
2413   \cs_new:cpn { lngx #1 logo } {
2414     \texorpdfstring {
2415       \lngxlogo [ #1 ]
2416     } {
2417       LinguisTiX - #1
2418     }
2419   }
2420 }
2421 </logos>
```

(End of definition for \lngxbaselogo and others. These functions are documented on page 9.)

```

2422 < *nfss >
2423 \ProvidesExplPackage{linguistix-nfss}
2424     {2025-06-19}
2425     {v0.5a}
2426     {%
2427         An extension to the core NFSS commands
2428         from the ‘LinguisTiX’ bundle.%
2429     }

```

I need a few temporary t_ls. I declare them here. As noted by the use of `__`, these are package-internal t_ls. Even though I don’t have any intention to change them, these are better not touched by the users.

```

2430
2431 \tl_new:N \l__lngx_normalfont_tmp_tl
2432 \tl_new:N \l__lngx_selectfont_tmp_tl
2433 \tl_new:N \l__lngx_family_tmp_tl
2434 \tl_new:N \l__lngx_nfss_tmp_tl

```

These t_ls are required for saving some values that are accessed later by the package as well as by the users.

```

2435
2436 \tl_new:N \l_lngx_current_encoding_tl
2437 \tl_new:N \l_lngx_current_meta_family_tl
2438 \tl_new:N \l_lngx_current_super_family_tl
2439 \tl_new:N \l_lngx_current_series_tl
2440 \tl_new:N \l_lngx_current_shape_tl

```

Here, I start the `begindocument/end` hook. After the document has started, a lot of initialisation can be assumed to have happened. I set some publicly available t_ls here.

```

\c_lngx_default_rmdefault_tl
\c_lngx_default_sfdefault_tl
\c_lngx_default_ttdefault_tl

```

```

2441
2442 \hook_gput_code:nnn { begindocument / end } { . } {
2443     \tl_const:Ne \c_lngx_default_rmdefault_tl { \rmdefault }
2444     \tl_const:Ne \c_lngx_default_sfdefault_tl { \sfdefault }
2445     \tl_const:Ne \c_lngx_default_ttdefault_tl { \ttdefault }

```

(End of definition for `\c_lngx_default_rmdefault_tl`, `\c_lngx_default_sfdefault_tl`, and `\c_lngx_default_ttdefault_tl`. These functions are documented on page 15.)

```

\l_lngx_current_encoding_tl
\l_lngx_current_meta_family_tl
\l_lngx_current_super_family_tl
\l_lngx_current_series_tl
\l_lngx_current_shape_tl

```

First, I set the value `default` for the initial super font family.

```

2446 \tl_set:Nn \l_lngx_current_super_family_tl { default }

```

The current encoding is saved in the relevant t_l.

```

2447 \tl_set:Ne \l_lngx_current_encoding_tl {
2448     \encodingdefault
2449 }

```

If the class is `beamer`, the font-family is automatically set to `sans`. Otherwise, mostly it is `serif`. Sadly, there is no public facing interface for confidently saying this, but as of now, this seems to be the picture. I check the current class and set the family t_l accordingly.

```

2450 \IfClassLoadedTF { beamer } {
2451     \tl_set:Ne \l_lngx_current_meta_family_tl { sf }
2452 } {
2453     \tl_set:Ne \l_lngx_current_meta_family_tl { rm }
2454 }

```

Here, the series and shape t1s are set to their defaults.

```

2455 \tl_set:Nc \l_lngx_current_series_tl { md }
2456 \tl_set:Nc \l_lngx_current_shape_tl { up }
2457 }

```

(End of definition for `\l_lngx_current_encoding_tl` and others. These functions are documented on page 15.)

The `\normalfont` command overrides the encoding. I trick the command by saving the encoding that was active before `\normalfont` in a temporary t1.

```

2458
2459 \hook_gput_code:nnn { cmd / normalfont / before } { . } {
2460 \tl_set:Nc \l__lngx_normalfont_tmp_tl { \f@encoding }
2461 }

```

After the processing of `\normalfont`, I equate the temporary t1 with the one that the package is tracking. This way, the effect of `\normalfont` remains unchanged, but we still save the values that were there before using it. Only encoding needs this special setting. Other attributes aren't reset by `\normalfont`.

```

2462
2463 \hook_gput_code:nnn { cmd / normalfont / after } { . } {
2464 \tl_set_eq:NN \l_lngx_current_encoding_tl
2465 \l__lngx_normalfont_tmp_tl
2466 \tl_clear:N \l__lngx_normalfont_tmp_tl
2467 }

```

Similar thing is done by `\selectfont` too. I repeat the code for that.

```

2468
2469 \hook_gput_code:nnn { cmd / selectfont / before } { . } {
2470 \tl_set:Nc \l__lngx_selectfont_tmp_tl { \f@encoding }
2471 }
2472
2473 \hook_gput_code:nnn { cmd / selectfont / after } { . } {
2474 \tl_set_eq:NN \l_lngx_current_encoding_tl
2475 \l__lngx_selectfont_tmp_tl
2476 \tl_clear:N \l__lngx_selectfont_tmp_tl
2477 }

```

Now, after each `\XXfamily` commands, I save the family name in the respective t1 for accessing later. All of these commands too reset the encoding. I repeat my trick for them too.

```

2478
2479 \hook_gput_code:nnn { cmd / rmfamily / before } { . } {
2480 \tl_set:Nc \l_lngx_current_meta_family_tl { rm }
2481 \tl_set:Nc \l__lngx_family_tmp_tl { \f@encoding }
2482 }
2483
2484 \hook_gput_code:nnn { cmd / rmfamily / after } { . } {
2485 \tl_set:Nc \l_lngx_current_meta_family_tl { rm }
2486 \tl_set_eq:NN \l_lngx_current_encoding_tl
2487 \l__lngx_family_tmp_tl
2488 \tl_clear:N \l__lngx_family_tmp_tl
2489 }
2490
2491 \hook_gput_code:nnn { cmd / sffamily / before } { . } {
2492 \tl_set:Nc \l_lngx_current_meta_family_tl { sf }

```



```

2493 \tl_set:Nc \l__lngx_family_tmp_tl { \f@encoding }
2494 }
2495
2496 \hook_gput_code:nnn { cmd / sffamily / after } { . } {
2497   \tl_set:Nn \l_lngx_current_meta_family_tl { sf }
2498   \tl_set_eq:NN \l_lngx_current_encoding_tl
2499     \l__lngx_family_tmp_tl
2500   \tl_clear:N \l__lngx_family_tmp_tl
2501 }
2502
2503 \hook_gput_code:nnn { cmd / ttfamily / before } { . } {
2504   \tl_set:Nn \l_lngx_current_meta_family_tl { tt }
2505   \tl_set:Nc \l__lngx_family_tmp_tl { \f@encoding }
2506 }
2507
2508 \hook_gput_code:nnn { cmd / ttfamily / after } { . } {
2509   \tl_set:Nn \l_lngx_current_meta_family_tl { tt }
2510   \tl_set_eq:NN \l_lngx_current_encoding_tl
2511     \l__lngx_family_tmp_tl
2512   \tl_clear:N \l__lngx_family_tmp_tl
2513 }

```

After the series commands, I save the series name in the `tl`. Note that, I don't use the traditional \LaTeX labels `m`, `bx` etc. Using, `md` and `bx` is more intuitive, plus they also can be used in the argument of `\use:c` directly.

```

2514
2515 \hook_gput_code:nnn { cmd / mdseries / after } { . } {
2516   \tl_set:Nn \l_lngx_current_series_tl { md }
2517 }
2518
2519 \hook_gput_code:nnn { cmd / bfseries / after } { . } {
2520   \tl_set:Nn \l_lngx_current_series_tl { bf }
2521 }

```

For shape related commands too, I save the names that are more closer to their respective commands.

```

2522
2523 \hook_gput_code:nnn { cmd / upshape / after } { . } {
2524   \tl_set:Nn \l_lngx_current_shape_tl { up }
2525 }
2526
2527 \hook_gput_code:nnn { cmd / itshape / after } { . } {
2528   \tl_set:Nn \l_lngx_current_shape_tl { it }
2529 }
2530
2531 \hook_gput_code:nnn { cmd / scshape / after } { . } {
2532   \tl_set:Nn \l_lngx_current_shape_tl { sc }
2533 }
2534
2535 \hook_gput_code:nnn { cmd / sscshape / after } { . } {
2536   \tl_set:Nn \l_lngx_current_shape_tl { ssc }
2537 }
2538
2539 \hook_gput_code:nnn { cmd / slshape / after } { . } {
2540   \tl_set:Nn \l_lngx_current_shape_tl { sl }

```

```

2541 }
2542
2543 \hook_gput_code:nnn { cmd / swshape / after } { . } {
2544   \tl_set:Nn \l_lngx_current_shape_tl { sw }
2545 }
2546
2547 \hook_gput_code:nnn { cmd / ulcshape / after } { . } {
2548   \tl_set:Nn \l_lngx_current_shape_tl { ulc }
2549 }
2550
2551 \hook_gput_code:nnn { cmd / ulcshape / after } { . } {
2552   \tl_set:Nn \l_lngx_current_shape_tl { #1 }
2553 }

```

`\lngx_if_encoding_p:n` I provide a conditional for checking the current encoding with the given argument.
`\lngx_if_encoding:nTF`

```

2554
2555 \prg_new_conditional:Nnn \lngx_if_encoding:n {
2556   p,
2557   T,
2558   F,
2559   TF
2560 } {
2561   \tl_if_eq:NnTF \l_lngx_current_encoding_tl { #1 } {
2562     \prg_return_true:
2563   } {
2564     \prg_return_false:
2565   }
2566 }
2567

```

(End of definition for `\lngx_if_encoding:nTF`. This function is documented on page 15.)

`\IfEncodingTF` For non-L^AT_EX₃ contexts, these simpler alternatives are provided.

```

2568 \IfEncodingT
2569 \IfEncodingF
2570 \cs_new_eq:NN \IfEncodingTF \lngx_if_encoding:nTF
2571 \cs_new_eq:NN \IfEncodingT \lngx_if_encoding:nT
2572 \cs_new_eq:NN \IfEncodingF \lngx_if_encoding:nF

```

(End of definition for `\IfEncodingTF`, `\IfEncodingT`, and `\IfEncodingF`. These functions are documented on page 11.)

`\lngx_if_meta_family_p:n` A conditional for checking the meta family with the given argument.

```

2572 \lngx_if_meta_family:nTF
2573 \prg_new_conditional:Nnn \lngx_if_meta_family:n {
2574   p,
2575   T,
2576   F,
2577   TF
2578 } {
2579   \tl_if_eq:NnTF \l_lngx_current_meta_family_tl { #1 } {
2580     \prg_return_true:
2581   } {
2582     \prg_return_false:
2583   }
2584 }

```

(End of definition for `\lngx_if_meta_family:nTF`. This function is documented on page 15.)

`\IfMetaFamilyTF` User-facing conditionals for meta family.
`\IfMetaFamilyT` 2585
`\IfMetaFamilyF` 2586 `\cs_new_eq:NN \IfMetaFamilyTF \lngx_if_meta_family:nTF`
2587 `\cs_new_eq:NN \IfMetaFamilyT \lngx_if_meta_family:nT`
2588 `\cs_new_eq:NN \IfMetaFamilyF \lngx_if_meta_family:nF`

(End of definition for `\IfMetaFamilyTF`, `\IfMetaFamilyT`, and `\IfMetaFamilyF`. These functions are documented on page 11.)

`\lngx_if_super_family_p:n` A conditional for checking the super family with the given argument.
`\lngx_if_super_family:nTF`

2589
2590 `\prg_new_conditional:Nnn \lngx_if_super_family:n {`
2591 `P,`
2592 `T,`
2593 `F,`
2594 `TF`
2595 `} {`
2596 `\tl_if_eq:NnTF \l_lngx_current_super_family_tl { #1 } {`
2597 `\prg_return_true:`
2598 `} {`
2599 `\prg_return_false:`
2600 `}`
2601 `}`

(End of definition for `\lngx_if_super_family:nTF`. This function is documented on page 15.)

`\IfSuperFamilyTF` User-facing conditionals for super family.
`\IfSuperFamilyT` 2602
`\IfSuperFamilyF` 2603 `\cs_new_eq:NN \IfSuperFamilyTF \lngx_if_super_family:nTF`
2604 `\cs_new_eq:NN \IfSuperFamilyT \lngx_if_super_family:nT`
2605 `\cs_new_eq:NN \IfSuperFamilyF \lngx_if_super_family:nF`

(End of definition for `\IfSuperFamilyTF`, `\IfSuperFamilyT`, and `\IfSuperFamilyF`. These functions are documented on page 11.)

`\lngx_if_series_p:n` A conditional for checking the current series with the given argument.
`\lngx_if_series:nTF`

2606
2607 `\prg_new_conditional:Nnn \lngx_if_series:n {`
2608 `P,`
2609 `T,`
2610 `F,`
2611 `TF`
2612 `} {`
2613 `\tl_if_eq:NnTF \l_lngx_current_series_tl { #1 } {`
2614 `\prg_return_true:`
2615 `} {`
2616 `\prg_return_false:`
2617 `}`
2618 `}`

(End of definition for `\lngx_if_series:nTF`. This function is documented on page 15.)

`\IfSeriesTF` Its user-side macros.

```

\IfSeriesT      2619
\IfSeriesF      2620 \cs_new_eq:NN \IfSeriesTF \lngx_if_series:nTF
                  2621 \cs_new_eq:NN \IfSeriesT  \lngx_if_series:nT
                  2622 \cs_new_eq:NN \IfSeriesF  \lngx_if_series:nF

```

(End of definition for `\IfSeriesTF`, `\IfSeriesT`, and `\IfSeriesF`. These functions are documented on page 11.)

`\lngx_if_shape_p:n` A conditional for checking the current shape with the current argument.

```

\lngx_if_shape:nTF 2623
                  2624 \prg_new_conditional:Nnn \lngx_if_shape:n {
                  2625   p,
                  2626   T,
                  2627   F,
                  2628   TF
                  2629 } {
                  2630   \tl_if_eq:NnTF \l_lngx_current_shape_tl { #1 } {
                  2631     \prg_return_true:
                  2632   } {
                  2633     \prg_return_false:
                  2634   }
                  2635 }

```

(End of definition for `\lngx_if_shape:nTF`. This function is documented on page 15.)

`\IfShapeTF` User-side macros for the same.

```

\IfShapeT      2636
\IfShapeF      2637 \cs_new_eq:NN \IfShapeTF \lngx_if_shape:nTF
                  2638 \cs_new_eq:NN \IfShapeT  \lngx_if_shape:nT
                  2639 \cs_new_eq:NN \IfShapeF  \lngx_if_shape:nF

```

(End of definition for `\IfShapeTF`, `\IfShapeT`, and `\IfShapeF`. These functions are documented on page 11.)

Now I will use the `\clist_map_inline:nn` technique for generating multiple conditionals of the same pattern. For that, I need a `cnn` variant of `\prg_new_conditional:Nnn` that I create with the following.

```

2640
2641 \cs_generate_variant:Nn \prg_new_conditional:Nnn { cnn }

```

`\lngx_if_meta_family_rm_p:` These are separate conditionals for `rm`, `sf` and `tt` families. They don't require arguments.
`\lngx_if_meta_family_rm:TF` No user side commands are provided for these.

```

\lngx_if_meta_family_sf_p: 2642
\lngx_if_meta_family_sf:TF 2643 \clist_map_inline:nn {
\lngx_if_meta_family_tt_p: 2644   rm,
\lngx_if_meta_family_tt:TF 2645   sf,
                             2646   tt
                             2647 } {
                             2648   \prg_new_conditional:cnn {
                             2649     lngx _ if _ meta _ family _ #1 :
                             2650   } {
                             2651     p, T, F, TF
                             2652   } {
                             2653     \tl_if_eq:NnTF \l_lngx_current_meta_family_tl { #1 } {
                             2654       \prg_return_true:

```

(End of definition for `\lngx_if_meta_family_rm:TF`, `\lngx_if_meta_family_sf:TF`, and `\lngx_if_meta_family_tt:TF`. These functions are documented on page 15.)

```

2660
2661 \clist_map_inline:nn {
2662     md,
2663     bf
2664 } {
2665     \prg_new_conditional:cnn { lngx _ if _ series _ #1 : } {
2666         p, T, F, TF
2667     } {
2668         \tl_if_eq:NnTF \l_lngx_current_series_tl { #1 } {
2669             \prg_return_true:
2670         } {
2671             \prg_return_false:
2672         }
2673     }
2674 }

```

```

2675
2676 \clist_map_inline:nn {
2677     up,
2678     it,
2679     sc,
2680     ssc,
2681     sl,
2682     sw,
2683     ulc
2684 } {
2685     \prg_new_conditional:cnn { lngx _ if _ shape _ #1 : } {
2686         p, T, F, TF
2687     } {
2688         \tl_if_eq:NnTF \l_lngx_current_shape_tl { #1 } {
2689             \prg_return_true:
2690         } {
2691             \prg_return_false:
2692         }
2693     }
2694 }

```

save the `rm`, `sf` and `tt` defaults of the new super font family. `\l__lngx_nfss_tmp_tl` is defined by the command that creates the super font family.

```

2695
2696 \clist_map_inline:nn {
2697     rm,
2698     sf,
2699     tt
2700 } {
2701     \keys_define:nn { lngx _ nfss } {
2702         #1
2703         .code:n          = {
2704             \tl_gclear_new:c {
2705                 g _ lngx _ \l__lngx_nfss_tmp_tl _ #1 default _ tl
2706             }
2707             \tl_gset:cn {
2708                 g _ lngx _ \l__lngx_nfss_tmp_tl _ #1 default _ tl
2709             } { ##1 }
2710         }
2711     }
2712 }

```

`\lngx_super_font_family:nn` I first set the temporary `tl` with the name of the super font family retrieved from the first argument.

```

2713
2714 \cs_new_protected:Npn \lngx_super_font_family:nn #1#2 {
2715     \tl_set:Nx \l__lngx_nfss_tmp_tl { #1 }

```

Now, I pass the second argument to the key-set I just defined. The temporary `tl` is cleared. This function comes with a user-side macro.

```

2716     \keys_set:nn { lngx _ nfss } { #2 }
2717     \tl_clear:N \l__lngx_nfss_tmp_tl
2718 }
2719
2720 \cs_set_eq:NN \superfontfamily
2721     \lngx_super_font_family:nn

```

(End of definition for `\lngx_super_font_family:nn` and `\superfontfamily`. These functions are documented on page 16.)

`\lngx_soft_super_font_family:nn` I set the `tl` that saves the current font family to the first argument.

```

2722
2723 \cs_new_protected:Npn \lngx_soft_super_font_family:nn #1#2 {
2724     \tl_set:Nx \l_lngx_current_super_family_tl { #1 }

```

I first check if the `tl`s for `rm`, `sf` and `tt` are empty or not. Only if they are not, I use their content in the respective `\XXdefault`. This makes the use of all the keys optional. Only the keys that the user has used are processed here.

```

2725     \clist_map_inline:nn {
2726         rm,
2727         sf,
2728         tt
2729     } {
2730         \tl_if_empty:cF { g _ lngx _ #1 _ ##1 default _ tl } {
2731             \cs_set:cpe { ##1 default } {

```

```

2732         \tl_use:c { g _ lngx _ #1 _ ##1 default _ tl }
2733     }
2734 }
2735 }

```

After setting the `\XXdefault`, I use the `\normalfont` to initialise the super font family.

```

2736 \normalfont

```

Now all the aspects are reset. But, we have them saved in our `tl`s. So now depending on the attributes that the user wants to retrieve, I call those attributes again. The second argument is (expected to be) a comma-separated list of all such attributes. Thus, we change the super font family, but retain the already active attributes. This command has a user-facing macro.

```

2737 \clist_map_inline:nn { #2 } {
2738     \str_case:nn { ##1 } {
2739         { encoding } {
2740             \exp_args:NV \fontencoding
2741                 \l_lngx_current_encoding_tl
2742         }
2743         { family } {
2744             \use:c {
2745                 \l_lngx_current_meta_family_tl family
2746             }
2747             \exp_args:NV \fontencoding
2748                 \l_lngx_current_encoding_tl
2749             \selectfont
2750         }
2751         { series } {
2752             \use:c {
2753                 \l_lngx_current_series_tl series
2754             }
2755         }
2756         { shape } {
2757             \use:c {
2758                 \l_lngx_current_shape_tl shape
2759             }
2760         }
2761     }
2762 }
2763 }
2764
2765 \cs_set_eq:NN \softsuperfontfamily
2766             \lngx_soft_super_font_family:nn

```

(End of definition for `\lngx_soft_super_font_family:nn` and `\softsuperfontfamily`. These functions are documented on page 16.)

`\lngx softer super font family:n`
`\softsuperfontfamily`

This function excludes the encoding and resets all the other attributes. It comes with a user-side macro.

```

2767
2768 \cs_new_protected:Npn \lngx softer super font family:n #1 {
2769     \lngx_soft_super_font_family:nn { #1 } {
2770         family,
2771         series,
2772         shape

```

```

2773 }
2774 }
2775
2776 \cs_set_eq:NN \softersuperfontfamily
2777 \lngx softer_super_font_family:n

```

(End of definition for `\lngx softer_super_font_family:n` and `\softersuperfontfamily`. These functions are documented on page 16.)

`\lngx softest_super_font_family:n`
`\softestsuperfontfamily` This function resets all the attributes. It is available as a user-side macro.

```

2778
2779 \cs_new_protected:Npn \lngx_softest_super_font_family:n #1 {
2780   \lngx_soft_super_font_family:nn { #1 } {
2781     encoding,
2782     family,
2783     series,
2784     shape
2785   }
2786 }
2787
2788 \cs_set_eq:NN \softestsuperfontfamily
2789 \lngx_softest_super_font_family:n

```

(End of definition for `\lngx_softest_super_font_family:n` and `\softestsuperfontfamily`. These functions are documented on page 16.)

`\lngx_soft_normal_font:n`
`\softnormalfont` Following the same logic, I now provide the command for resetting to the default super family, but retaining the active attributes. I provide a user-side macro for this.

```

2790
2791 \cs_new_protected:Npn \lngx_soft_normal_font:n #1 {
2792   \tl_set:Nc \l_lngx_current_super_family_tl { default }
2793   \clist_map_inline:nn {
2794     rm,
2795     sf,
2796     tt
2797   } {
2798     \cs_set:cpe { ##1 default } {
2799       \tl_use:c { c _ lngx _ default _ ##1 default _ tl }
2800     }
2801   }
2802   \normalfont
2803   \clist_map_inline:nn { #1 } {
2804     \str_case:nn { ##1 } {
2805       { encoding } {
2806         \exp_args:NV \fontencoding
2807           \l_lngx_current_encoding_tl
2808       }
2809       { family } {
2810         \use:c {
2811           \l_lngx_current_meta_family_tl family
2812         }
2813         \exp_args:NV \fontencoding
2814           \l_lngx_current_encoding_tl
2815       }
2816       \selectfont
2817     }
2818   }

```



```

2817     { series } {
2818       \use:c {
2819         \l_lngx_current_series_tl series
2820       }
2821     }
2822     { shape } {
2823       \use:c {
2824         \l_lngx_current_shape_tl shape
2825       }
2826     }
2827   }
2828 }
2829 }
2830
2831 \cs_set_eq:NN \softnormalfont \lngx_soft_normal_font:n

```

(End of definition for `\lngx_soft_normal_font:n` and `\softnormalfont`. These functions are documented on page 16.)

`\lngx_softer_normal_font:` This is a parallel to the ‘softer’ super family command for the default super family.
`\softernormalfont`

```

2832
2833 \cs_new_protected:Npn \lngx_softer_normal_font: {
2834   \lngx_soft_normal_font:n {
2835     family,
2836     series,
2837     shape
2838   }
2839 }
2840
2841 \cs_set_eq:NN \softernormalfont \lngx_softer_normal_font:

```

(End of definition for `\lngx_softer_normal_font:` and `\softernormalfont`. These functions are documented on page 16.)

`\lngx_softest_normal_font:` This is a parallel to the ‘softest’ super family command for the default super family.
`\softestnormalfont`

```

2842
2843 \cs_new_protected:Npn \lngx_softest_normal_font: {
2844   \lngx_soft_normal_font:n {
2845     encoding,
2846     family,
2847     series,
2848     shape
2849   }
2850 }
2851
2852 \cs_set_eq:NN \softestnormalfont \lngx_softest_normal_font:

```

(End of definition for `\lngx_softest_normal_font:` and `\softestnormalfont`. These functions are documented on page 16.)

`\CurrentEncoding` Lastly, we create the commands that print the current values of the font attributes and
`\CurrentMetaFamily` end the package.

```

2853 \cs_new:Npn \CurrentEncoding {
2854   \tl_use:N \l_lngx_current_encoding_tl
2855 }

```

```

2856 \cs_new:Npn \CurrentMetaFamily {
2857   \tl_use:N \l_lngx_current_meta_family_tl
2858 }
2859 \cs_new:Npn \CurrentSuperFamily {
2860   \tl_use:N \l_lngx_current_super_family_tl
2861 }
2862 \cs_new:Npn \CurrentSeries {
2863   \tl_use:N \l_lngx_current_series_tl
2864 }
2865 \cs_new:Npn \CurrentShape {
2866   \tl_use:N \l_lngx_current_shape_tl
2867 }
2868 \</nfss>

```

(End of definition for `\CurrentEncoding` and others. These functions are documented on page [II](#).)

References

- Brigham, R. (2004). *The elements of typographic style* (4th ed.). Point Roberts, WA: Hartley & Marks, Publishers.
- Munn, A., & Gregorio, E. (2023, December 5). *Expex fails with unicode-math. how to avoid the clash?* Retrieved June 19, 2025, from <https://tex.stackexchange.com/q/703094>

GNU Free Documentation License

Version 1.3, 3 November 2008

Copyright © 2000, 2001, 2002, 2007, 2008 Free Software Foundation, Inc.

<https://fsf.org/>

Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

Preamble

The purpose of this License is to make a manual, textbook, or other functional and useful document ‘free’ in the sense of freedom: to assure everyone the effective freedom to copy and redistribute it, with or without modifying it, either commercially or noncommercially. Secondly, this License preserves for the author and publisher a way to get credit for their work, while not being considered responsible for modifications made by others.

This License is a kind of ‘copyleft’, which means that derivative works of the document must themselves be free in the same sense. It complements the GNU General Public License, which is a copyleft license designed for free software.

We have designed this License in order to use it for manuals for free software, because free software needs free documentation: a free program should come with manuals providing the same freedoms that the software does. But this License is not limited to software manuals; it can be used for any textual work, regardless of subject matter or whether it is published as a printed book. We recommend this License principally for works whose purpose is instruction or reference.

I. APPLICABILITY AND DEFINITIONS

This License applies to any manual or other work, in any medium, that contains a notice placed by the copyright holder saying it can be distributed under the terms of this License. Such a notice grants a world-wide, royalty-free license, unlimited in duration, to use that work under the conditions stated herein. The ‘**Document**’, below, refers to any such manual or work. Any member of the public is a licensee, and is addressed as ‘**you**’. You accept the license if you copy, modify or distribute the work in a way requiring permission under copyright law.

A ‘**Modified Version**’ of the Document means any work containing the Document or a portion of it, either copied verbatim, or with modifications and/or translated into another language.

A ‘**Secondary Section**’ is a named appendix or a front-matter section of the Document that deals exclusively with the relationship of the publishers or authors of the Document to the Document’s overall subject (or to related matters) and contains nothing that could fall directly within that overall subject. (Thus, if the Document is in part a textbook of mathematics, a Secondary Section may not explain any mathematics.) The relationship could be a matter of historical connection with the subject or with related matters, or of legal, commercial, philosophical, ethical or political position regarding them.

The ‘**Invariant Sections**’ are certain Secondary Sections whose titles are designated, as being those of Invariant Sections, in the notice that says that the Document is released under this License. If a section does not fit the above definition of Secondary then it is

not allowed to be designated as Invariant. The Document may contain zero Invariant Sections. If the Document does not identify any Invariant Sections then there are none.

The ‘**Cover Texts**’ are certain short passages of text that are listed, as Front-Cover Texts or Back-Cover Texts, in the notice that says that the Document is released under this License. A Front-Cover Text may be at most 5 words, and a Back-Cover Text may be at most 25 words.

A ‘**Transparent**’ copy of the Document means a machine-readable copy, represented in a format whose specification is available to the general public, that is suitable for revising the document straightforwardly with generic text editors or (for images composed of pixels) generic paint programs or (for drawings) some widely available drawing editor, and that is suitable for input to text formatters or for automatic translation to a variety of formats suitable for input to text formatters. A copy made in an otherwise Transparent file format whose markup, or absence of markup, has been arranged to thwart or discourage subsequent modification by readers is not Transparent. An image format is not Transparent if used for any substantial amount of text. A copy that is not ‘Transparent’ is called ‘**Opaque**’.

Examples of suitable formats for Transparent copies include plain ASCII without markup, Texinfo input format, L^AT_EX input format, SGML or XML using a publicly available DTD, and standard-conforming simple HTML, PostScript or PDF designed for human modification. Examples of transparent image formats include PNG, XCF and JPG. Opaque formats include proprietary formats that can be read and edited only by proprietary word processors, SGML or XML for which the DTD and/or processing tools are not generally available, and the machine-generated HTML, PostScript or PDF produced by some word processors for output purposes only.

The ‘**Title Page**’ means, for a printed book, the title page itself, plus such following pages as are needed to hold, legibly, the material this License requires to appear in the title page. For works in formats which do not have any title page as such, ‘Title Page’ means the text near the most prominent appearance of the work’s title, preceding the beginning of the body of the text.

The ‘**publisher**’ means any person or entity that distributes copies of the Document to the public.

A section ‘**Entitled XYZ**’ means a named subunit of the Document whose title either is precisely XYZ or contains XYZ in parentheses following text that translates XYZ in another language. (Here XYZ stands for a specific section name mentioned below, such as ‘**Acknowledgements**’, ‘**Dedications**’, ‘**Endorsements**’, or ‘**History**’.) To ‘**Preserve the Title**’ of such a section when you modify the Document means that it remains a section ‘**Entitled XYZ**’ according to this definition.

The Document may include Warranty Disclaimers next to the notice which states that this License applies to the Document. These Warranty Disclaimers are considered to be included by reference in this License, but only as regards disclaiming warranties: any other implication that these Warranty Disclaimers may have is void and has no effect on the meaning of this License.

2. VERBATIM COPYING

You may copy and distribute the Document in any medium, either commercially or noncommercially, provided that this License, the copyright notices, and the license notice saying this License applies to the Document are reproduced in all copies, and that you add no other conditions whatsoever to those of this License. You may not use technical

measures to obstruct or control the reading or further copying of the copies you make or distribute. However, you may accept compensation in exchange for copies. If you distribute a large enough number of copies you must also follow the conditions in section 3.

You may also lend copies, under the same conditions stated above, and you may publicly display copies.

3. COPYING IN QUANTITY

If you publish printed copies (or copies in media that commonly have printed covers) of the Document, numbering more than 100, and the Document's license notice requires Cover Texts, you must enclose the copies in covers that carry, clearly and legibly, all these Cover Texts: Front-Cover Texts on the front cover, and Back-Cover Texts on the back cover. Both covers must also clearly and legibly identify you as the publisher of these copies. The front cover must present the full title with all words of the title equally prominent and visible. You may add other material on the covers in addition. Copying with changes limited to the covers, as long as they preserve the title of the Document and satisfy these conditions, can be treated as verbatim copying in other respects.

If the required texts for either cover are too voluminous to fit legibly, you should put the first ones listed (as many as fit reasonably) on the actual cover, and continue the rest onto adjacent pages.

If you publish or distribute Opaque copies of the Document numbering more than 100, you must either include a machine-readable Transparent copy along with each Opaque copy, or state in or with each Opaque copy a computer-network location from which the general network-using public has access to download using public-standard network protocols a complete Transparent copy of the Document, free of added material. If you use the latter option, you must take reasonably prudent steps, when you begin distribution of Opaque copies in quantity, to ensure that this Transparent copy will remain thus accessible at the stated location until at least one year after the last time you distribute an Opaque copy (directly or through your agents or retailers) of that edition to the public.

It is requested, but not required, that you contact the authors of the Document well before redistributing any large number of copies, to give them a chance to provide you with an updated version of the Document.

4. MODIFICATIONS

You may copy and distribute a Modified Version of the Document under the conditions of sections 2 and 3 above, provided that you release the Modified Version under precisely this License, with the Modified Version filling the role of the Document, thus licensing distribution and modification of the Modified Version to whoever possesses a copy of it. In addition, you must do these things in the Modified Version:

- A. Use in the Title Page (and on the covers, if any) a title distinct from that of the Document, and from those of previous versions (which should, if there were any, be listed in the History section of the Document). You may use the same title as a previous version if the original publisher of that version gives permission.
- B. List on the Title Page, as authors, one or more persons or entities responsible for authorship of the modifications in the Modified Version, together with at least five of the principal authors of the Document (all of its principal authors, if it has fewer than five), unless they release you from this requirement.

- C. State on the Title page the name of the publisher of the Modified Version, as the publisher.
- D. Preserve all the copyright notices of the Document.
- E. Add an appropriate copyright notice for your modifications adjacent to the other copyright notices.
- F. Include, immediately after the copyright notices, a license notice giving the public permission to use the Modified Version under the terms of this License, in the form shown in the Addendum below.
- G. Preserve in that license notice the full lists of Invariant Sections and required Cover Texts given in the Document's license notice.
- H. Include an unaltered copy of this License.
- I. Preserve the section Entitled 'History', Preserve its Title, and add to it an item stating at least the title, year, new authors, and publisher of the Modified Version as given on the Title Page. If there is no section Entitled 'History' in the Document, create one stating the title, year, authors, and publisher of the Document as given on its Title Page, then add an item describing the Modified Version as stated in the previous sentence.
- J. Preserve the network location, if any, given in the Document for public access to a Transparent copy of the Document, and likewise the network locations given in the Document for previous versions it was based on. These may be placed in the 'History' section. You may omit a network location for a work that was published at least four years before the Document itself, or if the original publisher of the version it refers to gives permission.
- K. For any section Entitled 'Acknowledgements' or 'Dedications', Preserve the Title of the section, and preserve in the section all the substance and tone of each of the contributor acknowledgements and/or dedications given therein.
- L. Preserve all the Invariant Sections of the Document, unaltered in their text and in their titles. Section numbers or the equivalent are not considered part of the section titles.
- M. Delete any section Entitled 'Endorsements'. Such a section may not be included in the Modified Version.
- N. Do not retitle any existing section to be Entitled 'Endorsements' or to conflict in title with any Invariant Section.
- O. Preserve any Warranty Disclaimers.

If the Modified Version includes new front-matter sections or appendices that qualify as Secondary Sections and contain no material copied from the Document, you may at your option designate some or all of these sections as invariant. To do this, add their titles to the list of Invariant Sections in the Modified Version's license notice. These titles must be distinct from any other section titles.

You may add a section Entitled 'Endorsements', provided it contains nothing but endorsements of your Modified Version by various parties—for example, statements of

peer review or that the text has been approved by an organization as the authoritative definition of a standard.

You may add a passage of up to five words as a Front-Cover Text, and a passage of up to 25 words as a Back-Cover Text, to the end of the list of Cover Texts in the Modified Version. Only one passage of Front-Cover Text and one of Back-Cover Text may be added by (or through arrangements made by) any one entity. If the Document already includes a cover text for the same cover, previously added by you or by arrangement made by the same entity you are acting on behalf of, you may not add another; but you may replace the old one, on explicit permission from the previous publisher that added the old one.

The author(s) and publisher(s) of the Document do not by this License give permission to use their names for publicity for or to assert or imply endorsement of any Modified Version.

5. COMBINING DOCUMENTS

You may combine the Document with other documents released under this License, under the terms defined in section 4 above for modified versions, provided that you include in the combination all of the Invariant Sections of all of the original documents, unmodified, and list them all as Invariant Sections of your combined work in its license notice, and that you preserve all their Warranty Disclaimers.

The combined work need only contain one copy of this License, and multiple identical Invariant Sections may be replaced with a single copy. If there are multiple Invariant Sections with the same name but different contents, make the title of each such section unique by adding at the end of it, in parentheses, the name of the original author or publisher of that section if known, or else a unique number. Make the same adjustment to the section titles in the list of Invariant Sections in the license notice of the combined work.

In the combination, you must combine any sections Entitled ‘History’ in the various original documents, forming one section Entitled ‘History’; likewise combine any sections Entitled ‘Acknowledgements’, and any sections Entitled ‘Dedications’. You must delete all sections Entitled ‘Endorsements’.

6. COLLECTIONS OF DOCUMENTS

You may make a collection consisting of the Document and other documents released under this License, and replace the individual copies of this License in the various documents with a single copy that is included in the collection, provided that you follow the rules of this License for verbatim copying of each of the documents in all other respects.

You may extract a single document from such a collection, and distribute it individually under this License, provided you insert a copy of this License into the extracted document, and follow this License in all other respects regarding verbatim copying of that document.

7. AGGREGATION WITH INDEPENDENT WORKS

A compilation of the Document or its derivatives with other separate and independent documents or works, in or on a volume of a storage or distribution medium, is called an ‘aggregate’ if the copyright resulting from the compilation is not used to limit the legal rights of the compilation’s users beyond what the individual works permit. When the

Document is included in an aggregate, this License does not apply to the other works in the aggregate which are not themselves derivative works of the Document.

If the Cover Text requirement of section 3 is applicable to these copies of the Document, then if the Document is less than one half of the entire aggregate, the Document's Cover Texts may be placed on covers that bracket the Document within the aggregate, or the electronic equivalent of covers if the Document is in electronic form. Otherwise they must appear on printed covers that bracket the whole aggregate.

8. TRANSLATION

Translation is considered a kind of modification, so you may distribute translations of the Document under the terms of section 4. Replacing Invariant Sections with translations requires special permission from their copyright holders, but you may include translations of some or all Invariant Sections in addition to the original versions of these Invariant Sections. You may include a translation of this License, and all the license notices in the Document, and any Warranty Disclaimers, provided that you also include the original English version of this License and the original versions of those notices and disclaimers. In case of a disagreement between the translation and the original version of this License or a notice or disclaimer, the original version will prevail.

If a section in the Document is Entitled 'Acknowledgements', 'Dedications', or 'History', the requirement (section 4) to Preserve its Title (section 1) will typically require changing the actual title.

9. TERMINATION

You may not copy, modify, sublicense, or distribute the Document except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense, or distribute it is void, and will automatically terminate your rights under this License.

However, if you cease all violation of this License, then your license from a particular copyright holder is reinstated (a) provisionally, unless and until the copyright holder explicitly and finally terminates your license, and (b) permanently, if the copyright holder fails to notify you of the violation by some reasonable means prior to 60 days after the cessation.

Moreover, your license from a particular copyright holder is reinstated permanently if the copyright holder notifies you of the violation by some reasonable means, this is the first time you have received notice of violation of this License (for any work) from that copyright holder, and you cure the violation prior to 30 days after your receipt of the notice.

Termination of your rights under this section does not terminate the licenses of parties who have received copies or rights from you under this License. If your rights have been terminated and not permanently reinstated, receipt of a copy of some or all of the same material does not give you any rights to use it.

10. FUTURE REVISIONS OF THIS LICENSE

The Free Software Foundation may publish new, revised versions of the GNU Free Documentation License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns. See <https://www.gnu.org/licenses/>.

Each version of the License is given a distinguishing version number. If the Document specifies that a particular numbered version of this License ‘or any later version’ applies to it, you have the option of following the terms and conditions either of that specified version or of any later version that has been published (not as a draft) by the Free Software Foundation. If the Document does not specify a version number of this License, you may choose any version ever published (not as a draft) by the Free Software Foundation. If the Document specifies that a proxy can decide which future versions of this License can be used, that proxy’s public statement of acceptance of a version permanently authorizes you to choose that version for the Document.

II. RELICENSING

‘Massive Multiauthor Collaboration Site’ (or ‘MMC Site’) means any World Wide Web server that publishes copyrightable works and also provides prominent facilities for anybody to edit those works. A public wiki that anybody can edit is an example of such a server. A ‘Massive Multiauthor Collaboration’ (or ‘MMC’) contained in the site means any set of copyrightable works thus published on the MMC site.

‘CC-BY-SA’ means the Creative Commons Attribution-Share Alike 3.0 license published by Creative Commons Corporation, a not-for-profit corporation with a principal place of business in San Francisco, California, as well as future copyleft versions of that license published by that same organization.

‘Incorporate’ means to publish or republish a Document, in whole or in part, as part of another Document.

An MMC is ‘eligible for relicensing’ if it is licensed under this License, and if all works that were first published under this License somewhere other than this MMC, and subsequently incorporated in whole or in part into the MMC, (1) had no cover texts or invariant sections, and (2) were thus incorporated prior to November 1, 2008.

The operator of an MMC Site may republish an MMC contained in the site under CC-BY-SA on the same site at any time before August 1, 2009, provided the MMC is eligible for relicensing.

ADDENDUM: How to use this License for your documents

To use this License in a document you have written, include a copy of the License in the document and put the following copyright and license notices just after the title page:

Copyright © YEAR YOUR NAME.

Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.3 or any later version published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts. A copy of the license is included in the section entitled ‘GNU Free Documentation License’.

If you have Invariant Sections, Front-Cover Texts and Back-Cover Texts, replace the ‘with ... Texts.’ line with this:

with the Invariant Sections being LIST THEIR TITLES, with the Front-Cover Texts being LIST, and with the Back-Cover Texts being LIST.

If you have Invariant Sections without Cover Texts, or some other combination of the three, merge those two alternatives to suit the situation.

If your document contains nontrivial examples of program code, we recommend releasing these examples in parallel under your choice of free software license, such as the GNU General Public License, to permit their use in free software.