

The LINGUISṬIX bundle

निरंजन

May 29, 2025 (v0.2)

🏠 <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-ipa	7
2	Planned	4		Interface... 13; Implementation... 41	
3	Funding	4	8	LINGUISṬIX-Logos	9
4	Acknowledgements	4		Interface... 14; Implementation... 66	
5	LINGUISṬIX-BASE	5	9	LINGUISṬIX-NFSS	9
	Interface... 12; Implementation... 18			Interface... 15; Implementation... 68	
6	LINGUISṬIX-FONTS	5		GNU Free Documentation License	80
	Interface... 13; Implementation... 19				

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 [ɑ], 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 7.

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 [g̱g̱].

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 $\text{\textcolor{violet}{T}}\text{\textcolor{violet}{I}}\text{\textcolor{violet}{X}}$ -BASE

$\text{\textcolor{violet}{L}}\text{\textcolor{violet}{A}}\text{\textcolor{violet}{T}}\text{\textcolor{violet}{E}}\text{\textcolor{violet}{X}}_3\text{-interface}$ | **Implementation**

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

$\text{\textbackslash linguistix}$ $\{ \langle key-value-list \rangle \}$

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 $\text{\textcolor{violet}{T}}\text{\textcolor{violet}{I}}\text{\textcolor{violet}{X}}$ and use the $\text{\textbackslash linguistix}$ command. The only exception to this is LINGUIS $\text{\textcolor{violet}{T}}\text{\textcolor{violet}{I}}\text{\textcolor{violet}{X}}$ -NFSS. We will see how it is different in its section.

6 LINGUIS $\text{\textcolor{violet}{T}}\text{\textcolor{violet}{I}}\text{\textcolor{violet}{X}}$ -FONTS

$\text{\textcolor{violet}{L}}\text{\textcolor{violet}{A}}\text{\textcolor{violet}{T}}\text{\textcolor{violet}{E}}\text{\textcolor{violet}{X}}_3\text{-interface}$ | **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 $\text{\textcolor{violet}{T}}\text{\textcolor{violet}{I}}\text{\textcolor{violet}{X}}$ -IPA separately.

Antonis suggested a typographic enhancement for the logo of $\text{\textcolor{violet}{L}}\text{\textcolor{violet}{A}}\text{\textcolor{violet}{T}}\text{\textcolor{violet}{E}}\text{\textcolor{violet}{X}}$. 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.

$\text{\textbackslash LaTeX}$ $\text{\textcolor{violet}{L}}\text{\textcolor{violet}{A}}\text{\textcolor{violet}{T}}\text{\textcolor{violet}{E}}\text{\textcolor{violet}{X}}$
 $\text{\textbackslash ogLaTeX}$ $\text{\textcolor{violet}{L}}\text{\textcolor{violet}{A}}\text{\textcolor{violet}{T}}\text{\textcolor{violet}{E}}\text{\textcolor{violet}{X}}$

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

6.1 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 $\text{\textcolor{violet}{T}}\text{\textcolor{violet}{I}}\text{\textcolor{violet}{X}}$ -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.

<hr/> old style numbers	= $\{(\textit{truth value})\}$	true false
old style one	= $\{(\textit{truth value})\}$	true false

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

<hr/> newcm	
newcm sans	
newcm mono	
newcm regular	
newcm regular sans	
newcm regular mono	

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.

6.2 Maths

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

<hr/> math	= $\{(\textit{math font})\}$
math features	= $\{(\textit{math font features})\}$
math bold	= $\{(\textit{bold math font})\}$
math bold features	= $\{(\textit{bold math font features})\}$

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.

<hr/> bourbaki's empty set	= $\{(\textit{truth value})\}$	true false
----------------------------	--------------------------------	--------------

In (LA)TEX, 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: ‘ \emptyset ’. In order to change this behaviour, one may use this key and set it to false for getting the slashed-zero of original (LA)TEX. Hail plumbers union, *IYKYK!* ;-)

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

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 $\overline{\text{C}}$ I $\overline{\text{X}}$ -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{aɪ phi: eɪ}` \longrightarrow [a_ɪ p^hi: e_ɪ] whereas the starred version prints it in slashes for phonemic transcription, e.g.: `\ipatext*{aɪ phi: eɪ}` \longrightarrow /a_ɪ p^hi: e_ɪ/.

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_ɪ p^hi: e_ɪ] and /a_ɪ p^hi: e_ɪ/ respectively:

```
{\lngxipa [aɪ phi: eɪ]}
{\lngxipa /aɪ phi: eɪ/}
```

```
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 $\overline{\text{C}}$ I $\overline{\text{X}}$ -FONTS and LINGUIS $\overline{\text{C}}$ I $\overline{\text{X}}$ -ipa.

Family	LINGUIS $\overline{\text{C}}$ I $\overline{\text{X}}$ -FONTS	LINGUIS $\overline{\text{C}}$ I $\overline{\text{X}}$ -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
<i>Continued on the next page...</i>		

Family	LINGUISCI-X-FONTS	LINGUISCI-X-IPA
	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
	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

Continued on the next page...

Family	LINGUIS \mathcal{T} I \mathbf{X} -FONTS	LINGUIS \mathcal{T} I \mathbf{X} -IPA
	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 LINGUIS \mathcal{T} I \mathbf{X} -FONTS and LINGUIS \mathcal{T} I \mathbf{X} -IPA

8 LINGUIS \mathcal{T} I \mathbf{X} -LOGOS

L^AT_EX₃-interface | Implementation

This is a small package that provides commands for printing logos of the LINGUIS \mathcal{T} I \mathbf{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 LINGUIS \mathcal{T} I \mathbf{X} bundle is printed with this command, e.g., `\lngxlogo{fonts} → LINGUIS \mathcal{T} I \mathbf{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>	★	LINGUIS \mathcal{T} I \mathbf{X}
<code>\lngxbaselogo</code>	★	LINGUIS \mathcal{T} I \mathbf{X} -BASE
<code>\lngxfontlogo</code>	★	LINGUIS \mathcal{T} I \mathbf{X} -FONTS
<code>\lngxipalogo</code>	★	LINGUIS \mathcal{T} I \mathbf{X} -IPA
<code>\lngxlogoslogo</code>	★	LINGUIS \mathcal{T} I \mathbf{X} -LOGOS
<code>\lngxnfsslogo</code>	★	LINGUIS \mathcal{T} I \mathbf{X} -NFSS

9 LINGUIS \mathcal{T} I \mathbf{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 LINGUISTIX 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. LINGUISTIX-NFFS 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>	* <code>{\langle encoding \rangle}{\langle true code \rangle}{\langle false code \rangle}</code>
<code>\IfEncodingT</code>	* <code>{\langle encoding \rangle}{\langle true code \rangle}</code>
<code>\IfEncodingF</code>	* <code>{\langle encoding \rangle}{\langle false code \rangle}</code>
<code>\CurrentEncoding</code>	* If the current encoding matches with the given <code>\langle encoding \rangle</code> , it selects the true branch; false otherwise. The <code>\CurrentEncoding</code> macro expands to the current encoding.
<hr/>	
<code>\IfMetaFamilyTF</code>	* <code>{\langle meta family \rangle}{\langle true code \rangle}{\langle false code \rangle}</code>
<code>\IfMetaFamilyT</code>	* <code>{\langle meta family \rangle}{\langle true code \rangle}</code>
<code>\IfMetaFamilyF</code>	* <code>{\langle meta family \rangle}{\langle false code \rangle}</code>
<code>\CurrentMetaFamily</code>	* If the current meta family matches with the given <code>\langle meta family \rangle</code> , it selects the true branch; false otherwise. The <code>\CurrentMetaFamily</code> macro expands to the current meta family.
<hr/>	
<code>\IfSuperFamilyTF</code>	* <code>{\langle super family \rangle}{\langle true code \rangle}{\langle false code \rangle}</code>
<code>\IfSuperFamilyT</code>	* <code>{\langle super family \rangle}{\langle true code \rangle}</code>
<code>\IfSuperFamilyF</code>	* <code>{\langle super family \rangle}{\langle false code \rangle}</code>
<code>\CurrentSuperFamily</code>	* If the current super family matches with the given <code>\langle super family \rangle</code> , it selects the true branch; false otherwise. The <code>\CurrentSuperFamily</code> macro expands to the current super family.
<hr/>	
<code>\IfSeriesTF</code>	* <code>{\langle series \rangle}{\langle true code \rangle}{\langle false code \rangle}</code>
<code>\IfSeriesT</code>	* <code>{\langle series \rangle}{\langle true code \rangle}</code>
<code>\IfSeriesF</code>	* <code>{\langle series \rangle}{\langle false code \rangle}</code>
<code>\CurrentSeries</code>	* If the current series matches with the given <code>\langle series \rangle</code> , it selects the true branch and false otherwise. The <code>\CurrentSeries</code> macro expands to the current series.
<hr/>	
<code>\IfShapeTF</code>	* <code>{\langle shape \rangle}{\langle true code \rangle}{\langle false code \rangle}</code>
<code>\IfShapeT</code>	* <code>{\langle shape \rangle}{\langle true code \rangle}</code>
<code>\IfShapeF</code>	* <code>{\langle shape \rangle}{\langle false code \rangle}</code>
<code>\CurrentShape</code>	* If the current series matches with the given <code>\langle shape \rangle</code> , it selects the true branch and false otherwise. The <code>\CurrentShape</code> macro expands to the current shape.
<hr/>	
<code>\superfontfamily</code>	<code>{\langle family id \rangle}{\langle rm=\langle rm nfss \rangle,sf=\langle sf nfss \rangle,tt=\langle tt nfss \rangle \rangle}</code>
	Every super font family has a <code>\langle family id \rangle</code> , even the default one (i.e., <code>default</code>). This command creates a super family with the given <code>\langle family id \rangle</code> s. The <code>\langle meta family keys \rangle</code> argument accepts a list of specific keys, <code>rm</code> , <code>sf</code> and <code>tt</code> . They take the NFSS family names of these meta families as arguments. One may define a font with, say, <code>\newfontfamily</code> , pass the <code>NFSSkeys={\langle key \rangle}</code> option to it and use the <code>\langle key \rangle</code> in the suitable <code>\langle meta family key \rangle</code> . Note that using all these keys is <i>not</i> 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.

<code>\g_lngx_old_style_bool</code>	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., ‘∅’) is asked by the user.
<code>\g_lngx_old_style_one_bool</code>	
<code>\g_lngx_bourbaki_bool</code>	

```

\lngx_set_main_font:nm {\features} {\font}
\lngx_set_main_font:ee {\features} {\font}
\lngx_set_sans_font:nm {\features} {\font}
\lngx_set_sans_font:ee {\features} {\font}

```

```

\lngx_set_mono_font:nm
\lngx_set_mono_font:ee
\lngx_set_math_font:nm
\lngx_set_math_font:ee

```

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., `\setmainfont`, `\setsansfont`, `\setmonofont` and `\setmathfont`. The `\features` are passed to the optional argument and the `\font` is passed to the mandatory argument of the respective command from the aforementioned list.

LINGUISŢIX-ipa

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

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

<code>\lngx_set_main_ipa_font:nm</code>	These functions set the IPA fonts for the serif variants. The <code>\font</code> is set with <code>\features</code> 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> .
<code>\lngx_set_main_ipa_font:ee</code>	
<code>\lngx_main_ipa:</code> <code>lngx_ipa_rm_nfss</code>	

<code>\lngx_set_sans_ipa_font:nm</code>	These functions set the IPA fonts for the sans variants. The <code>\font</code> is set with <code>\features</code> 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> .
<code>\lngx_set_sans_ipa_font:ee</code>	
<code>\lngx_sans_ipa:</code> <code>lngx_ipa_sf_nfss</code>	

<code>\lngx_set_mono_ipa_font:nm</code>	These functions set the IPA fonts for the mono variants. The <code>\font</code> is set with <code>\features</code> 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> .
<code>\lngx_set_mono_ipa_font:ee</code>	
<code>\lngx_mono_ipa:</code> <code>lngx_ipa_tt_nfss</code>	

<code>\lngx_ipa:</code> <code>lngx_ipa</code>	The <code>\lngx_ipa:</code> command loads the super family <code>lngx_ipa</code> (see the documentation of LINGUISŢIX-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 LINGUISŢIX-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 `LINGUISCIX-FONTS` and `LINGUISCIX-ipa`

LINGUISCIX-LOGOS

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

There are only two \LaTeX 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 $\overline{\text{C}}$ IX-NFSS

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

This subsection discusses the programming interface LINGUIS $\overline{\text{C}}$ IX-NFSS provides.

`\c_lngx_default_rmdefault_tl` * These t_ls 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 t_ls 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 * {\true code}{\false code}
\lngx_if_shape_it_p: *
\lngx_if_shape_it:TF * {\true code}{\false code}
\lngx_if_shape_sc_p: *
\lngx_if_shape_sc:TF * {\true code}{\false code}
\lngx_if_shape_ssc_p: *
\lngx_if_shape_ssc:TF * {\true code}{\false code}
\lngx_if_shape_sl_p: *
\lngx_if_shape_sl:TF * {\true code}{\false code}
\lngx_if_shape_sw_p: *
\lngx_if_shape_sw:TF * {\true code}{\false code}
\lngx_if_shape_ulc_p: *
\lngx_if_shape_ulc:TF * {\true code}{\false code}

```

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 {\family id} {\rm=\{rm nfss\},sf=\{sf nfss\},tt=\{tt nfss\}}

```

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                      {2025-05-29}
4                      {v0.2}
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             {2025-05-29}
29             {v0.2}
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**TiX**-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.)

Package essentials first.

```

41  \font
42  \ProvidesExplPackage{linguistix-fonts}
43      {2025-05-29}
44      {v0.2}
45      {%
46          The font-assistant package of the
47          'LinguisTiX' bundle.%
48      }

```

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

```

49
50  \IfPackageLoadedF { linguistix-base } {
51      \RequirePackage { linguistix-base }
52  }
53
54  \IfPackageLoadedF { unicode-math } {
55      \RequirePackage { unicode-math }
56  }

```

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

```

57
58  \NewCommandCopy \ogLaTeX \LaTeX
59
60  \RenewDocumentCommand \LaTeX { } {%
61      L\kern-.81ex\relax
62      \raisebox{.6ex}{\textsc{a}}\kern-.23ex\relax
63      \hbox{T}\kern-.4ex\relax
64      \raisebox{-.5ex}{E}\kern-.3ex\relax
65      X%
66  }

```

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

old style numbers
\g_lngx_old_style_bool
old style one
\g_lngx_old_style_one_bool
bourbaki's empty set
\g_lngx_bourbaki_bool

I use the .bool_gset:N key-type of l3keys for developing these boolean keys.

```

67
68  \keys_define:nn { lngx _ keys } {
69      old~ style~ numbers
70          .bool_gset:N          = {
71              \g_lngx_old_style_bool
72          },
73      old~ style~ one
74          .bool_gset:N          = {
75              \g_lngx_old_style_one_bool
76          },
77      bourbaki's~ empty~ set
78          .bool_gset:N          = {
79              \g_lngx_bourbaki_bool
80          }
81  }

```

(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>.)

```

82
83 \clist_map_inline:nn {
84   upright,
85   bold~ upright,
86   italic,
87   bold~ italic,
88   slanted,
89   bold~ slanted,
90   swash,
91   bold~ swash,
92   small~ caps
93 } {

```

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.

```

94 \tl_set:Nn \l_tmpa_tl { #1 }
95 \tl_replace_all:Nnn \l_tmpa_tl { ~ } { _ }
96 \tl_gclear_new:c {
97   g _ lngx _ text _ \l_tmpa_tl _ features _ tl
98 }

```

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`.

```

99 \keys_define:nn { lngx _ keys } {
100   text~ #1
101   .tl_gset_e:c = {
102     g _ lngx _ text _ \l_tmpa_tl _ tl
103   },

```

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.

```

104   text~ #1~ features
105   .code:n = {
106     \tl_set:Nn \l_tmpb_tl { #1 }
107     \tl_replace_all:Nnn \l_tmpb_tl { ~ } { _ }
108     \tl_put_right:ce {
109       g _ lngx _ text _ \l_tmpb_tl _ features _ tl
110     } { ##1 , }

```

Lastly, we clear the temporary `tl`s.

```

111   \tl_clear:N \l_tmpb_tl
112 }
113 }
114 \tl_clear:N \l_tmpa_tl
115 }

```

(End of definition for `text upright` and others. These functions are documented on page 7.)

```

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.

```

II6
II7 \clist_map_inline:nn {
II8   upright,
II9   bold~ upright,
I20   italic,
I21   bold~ italic,
I22   slanted,
I23   bold~ slanted,
I24   swash,
I25   bold~ swash,
I26   small~ caps
I27 } {
I28   \tl_set:Nn \l_tmpa_tl { #1 }
I29   \tl_replace_all:Nnn \l_tmpa_tl { ~ } { _ }
I30   \tl_gclear_new:c {
I31     g _ lngx _ text _ sans _ \l_tmpa_tl _ features _ tl
I32   }
I33   \keys_define:nn { lngx _ keys } {
I34     text~ sans~ #1
I35     .tl_gset_e:c          = {
I36       g _ lngx _ text _ sans _ \l_tmpa_tl _ tl
I37     },
I38     text~ sans~ #1~ features
I39     .code:n              = {
I40       \tl_set:Nn \l_tmpb_tl { #1 }
I41       \tl_replace_all:Nnn \l_tmpb_tl { ~ } { _ }
I42       \tl_put_right:ce {
I43         g _ lngx _ text _ sans _ \l_tmpb_tl _ features _ tl
I44       } { ##1 , }
I45       \tl_clear:N \l_tmpb_tl
I46     }
I47   }
I48   \tl_clear:N \l_tmpa_tl
I49 }

```

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

```

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

```

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

```

150
151 \clist_map_inline:nn {
152   upright,
153   bold~ upright,
154   italic,
155   bold~ italic,
156   slanted,
157   bold~ slanted,
158   swash,
159   bold~ swash,
160   small~ caps
161 } {
162   \tl_set:Nn \l_tmpa_tl { #1 }
163   \tl_replace_all:Nnn \l_tmpa_tl { ~ } { _ }
164   \tl_gclear_new:c {
165     g _ lngx _ text _ mono _ \l_tmpa_tl _ features _ tl
166   }
167   \keys_define:nn { lngx _ keys } {
168     text~ mono~ #1
169     .tl_gset_e:c          = {
170       g _ lngx _ text _ mono _ \l_tmpa_tl _ tl
171     },
172     text~ mono~ #1~ features
173     .code:n              = {
174       \tl_set:Nn \l_tmpb_tl { #1 }
175       \tl_replace_all:Nnn \l_tmpb_tl { ~ } { _ }
176       \tl_put_right:ce {
177         g _ lngx _ text _ mono _ \l_tmpb_tl _ features _ tl
178       } { ##1 , }
179       \tl_clear:N \l_tmpb_tl
180     }
181   }
182   \tl_clear:N \l_tmpa_tl
183 }

```

(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.

```

184
185 \keys_define:nn { lngx _ keys } {
186   math
187   .tl_gset_e:c          = {
188     g _ lngx _ math _ tl
189   },
190   math~ features
191   .tl_gset_e:c          = {
192     g _ lngx _ math _ features _ tl
193   },
194   math~ bold
195   .tl_gset_e:c          = {
196     g _ lngx _ math _ bold _ tl
197   },
198   math~ bold~ features
199   .code:n               = {
200     \tl_put_right:ce {
201       g _ lngx _ math _ bold _ features _ tl
202     } { #1 }
203   }
204 }

```

*(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.

```

205
206 \keys_define:nn { lngx _ keys } {
207   newcm
208   .meta:n              = {
209     text~
210     upright             = {
211       NewCM10-Book.otf
212     },
213     text~
214     bold~ upright       = {
215       NewCM10-Bold.otf
216     },
217     text~
218     italic              = {
219       NewCM10-BookItalic.otf
220     },
221     text~
222     bold~ italic        = {
223       NewCM10-BoldItalic.otf
224     },
225     math                = {
226       NewCMMath-Book.otf
227     },
228     math~ bold          = {
229       NewCMMath-Bold.otf
230     },
231     text~

```

```

232 sans~ upright          = {
233     NewCMSans10-Book.otf
234 },
235 text~
236 sans~ bold~ upright    = {
237     NewCMSans10-Bold.otf
238 },
239 text~
240 sans~ italic           = {
241     NewCMSans10-BookOblique.otf
242 },
243 text~
244 sans~ bold~ italic     = {
245     NewCMSans10-BoldOblique.otf
246 },
247 text~
248 mono~ upright          = {
249     NewCMMono10-Book.otf
250 },
251 text~
252 mono~ bold~ upright    = {
253     NewCMMono10-Bold.otf
254 },
255 text~
256 mono~ italic           = {
257     NewCMMono10-BookItalic.otf
258 },
259 text~
260 mono~ bold~ italic     = {
261     NewCMMono10-BoldOblique.otf
262 }
263 }
264 }

```

(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.

```

265
266 \keys_define:nn { lngx _ keys } {
267     newcm~ sans
268     .meta:n          = {
269         text~
270         upright      = {
271             NewCMSans10-Book.otf
272         },
273         text~
274         bold upright = {
275             NewCMSans10-Bold.otf
276         },
277         text~
278         italic       = {
279             NewCMSans10-BookOblique.otf
280         },
281         text~

```



```

282     bold~ italic          = {
283         NewCMSans10-BoldOblique.otf
284     }
285 }
286 }

```

(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.

```

287
288 \keys_define:nn { lngx _ keys } {
289     newcm~ mono
290     .meta:n          = {
291         text~
292         upright       = {
293             NewCMMono10-Book.otf
294         },
295         text~
296         bold upright  = {
297             NewCMMono10-Bold.otf
298         },
299         text~
300         italic        = {
301             NewCMMono10-BookItalic.otf
302         },
303         text~
304         bold~ italic  = {
305             NewCMMono10-BoldOblique.otf
306         }
307     }
308 }

```

(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.

```

309
310 \keys_define:nn { lngx _ keys } {
311     newcm~ regular
312     .meta:n          = {
313         text~
314         upright       = {
315             NewCM10-Regular.otf
316         },
317         text~
318         bold~ upright = {
319             NewCM10-Bold.otf
320         },
321         text~
322         italic        = {
323             NewCM10-Italic.otf
324         },
325         text~
326         bold~ italic  = {

```

```

327     NewCM10-BoldItalic.otf
328 },
329 math                = {
330     NewCMMath-Regular.otf
331 },
332 math~ bold          = {
333     NewCMMath-Bold.otf
334 },
335 text~
336 sans~ upright       = {
337     NewCMSans10-Regular.otf
338 },
339 text~
340 sans~ bold~ upright = {
341     NewCMSans10-Bold.otf
342 },
343 text~
344 sans~ italic        = {
345     NewCMSans10-Oblique.otf
346 },
347 text~
348 sans~ bold~ italic  = {
349     NewCMSans10-BoldOblique.otf
350 },
351 text~
352 mono~ upright       = {
353     NewCMMono10-Regular.otf
354 },
355 text~
356 mono~ bold~ upright = {
357     NewCMMono10-Bold.otf
358 },
359 text~
360 mono~ italic        = {
361     NewCMMono10-Italic.otf
362 },
363 text~
364 mono~ bold~ italic  = {
365     NewCMMono10-Bold.otf
366 }
367 }
368 }

```

(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.

```

369
370 \keys_define:nn { lngx _ keys } {
371   newcm~ regular~ sans
372   .meta:n                = {
373     text~
374     upright               = {
375       NewCMSans10-Regular.otf

```

```

376 },
377 text~
378 bold~ upright          = {
379   NewCMSans10-Bold.otf
380 },
381 text~
382 italic                 = {
383   NewCMSans10-Oblique.otf
384 },
385 text~
386 bold~ italic          = {
387   NewCMSans10-BoldOblique.otf
388 }
389 }
390 }

```

(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.

```

391
392 \keys_define:nn { lngx _ keys } {
393   newcm~ regular~ mono
394   .meta:n          = {
395     text~
396     upright         = {
397       NewCMMono10-Regular.otf
398     },
399     text~
400     bold~ upright   = {
401       NewCMMono10-Bold.otf
402     },
403     text~
404     italic          = {
405       NewCMMono10-Italic.otf
406     },
407     text~
408     bold~ italic    = {
409       NewCMMono10-Bold.otf
410     }
411   }
412 }

```

(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.

```

413
414 \lngx_set_keys:n {
415   newcm,

```

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

```

416   bourbaki's~ empty~ set,

```

Lastly we load the old `style numbers` boolean.

```

417   old~ style~ numbers
418 }

```

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.

```

419
420 \sys_if_engine luatex:T {
421   \lngx_set_keys:n {
422     text~
423     upright~ features      = {
424       Renderer             = { HarfBuzz }
425     },
426     text~ sans~
427     upright~ features      = {
428       Renderer             = { HarfBuzz }
429     },
430     text~ mono~
431     upright~ features      = {
432       Renderer             = { HarfBuzz }
433     }
434   }
435 }

```

`\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.

```

436
437 \cs_new_protected:Npn \lngx_set_main_font:nn #1#2 {
438   \setmainfont [ #1 ] { #2 }
439 }
440
441 \cs_new_protected:Npn \lngx_set_sans_font:nn #1#2 {
442   \setsansfont [ #1 ] { #2 }
443 }
444
445 \cs_new_protected:Npn \lngx_set_mono_font:nn #1#2 {
446   \setmonofont [ #1 ] { #2 }
447 }
448
449 \cs_new_protected:Npn \lngx_set_math_font:nn #1#2 {
450   \setmathfont [ #1 ] { #2 }
451 }
452
453 \cs_generate_variant:Nn \lngx_set_main_font:nn { ee }
454 \cs_generate_variant:Nn \lngx_set_sans_font:nn { ee }
455 \cs_generate_variant:Nn \lngx_set_mono_font:nn { ee }
456 \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.

```

457
458 \hook_gput_code:nnn { begindocument / before } { . } {
459   \tl_if_in:cnT {
460     g _ lngx _ text _ upright _ tl
461   } { 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.

```

462   \tl_if_in:cnTF {
463     g _ lngx _ text _ upright _ tl
464   } { Book } {
465     \lngx_set_keys:n {
466       text~
467       upright~ features      = {
468         SizeFeatures         = {
469           {
470             Size              = {-8},
471             Font              = {
472               NewCM08-Book.otf
473             }
474           },
475           {
476             Size              = {8-},
477             Font              = {
478               NewCM10-Book.otf
479             }
480           }
481         }
482       }
483     }

```

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

```

484   } {
485     \lngx_set_keys:n {
486       text~
487       upright~ features      = {
488         SizeFeatures         = {
489           {
490             Size              = {-8},
491             Font              = {
492               NewCM08-Regular.otf
493             }
494           },
495           {
496             Size              = {8-},
497             Font              = {
498               NewCM10-Regular.otf
499             }
500           }
501         }
502       }

```

```

503     }
504   }
505 }

```

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.

```

506 \tl_if_in:cnT {
507   g _ lngx _ text _ upright _ tl
508 } { NewCMSans } {
509   \tl_if_in:cnTF {
510     g _ lngx _ text _ upright _ tl
511   } { Book } {
512     \lngx_set_keys:n {
513       text~
514       upright~ features      = {
515         SizeFeatures        = {
516           {
517             Size             = {-8},
518             Font              = {
519               NewCMSans08-Book.otf
520             }
521           },
522           {
523             Size             = {8-},
524             Font              = {
525               NewCMSans10-Book.otf
526             }
527           }
528         }
529       }
530     }
531   } {
532     \lngx_set_keys:n {
533       text~
534       upright~ features      = {
535         SizeFeatures        = {
536           {
537             Size             = {-8},
538             Font              = {
539               NewCMSans08-Regular.otf
540             }
541           },
542           {
543             Size             = {8-},
544             Font              = {
545               NewCMSans10-Regular.otf
546             }
547           }
548         }
549       }

```

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.

```

550     }
551   }
552 }

```

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

```

553 \tl_if_in:cnT {
554   g _ lngx _ text _ italic _ tl
555 } { NewCM } {
556   \tl_if_in:cnTF {
557     g _ lngx _ text _ italic _ tl
558   } { Book } {
559     \lngx_set_keys:n {
560       text~
561       italic~ features      = {
562         SizeFeatures        = {
563           {
564             Size              = {-8},
565             Font              = {
566               NewCM08-BookItalic.otf
567             }
568           },
569           {
570             Size              = {8-},
571             Font              = {
572               NewCM10-BookItalic.otf
573             }
574           }
575         }
576       }
577     }
578   } {
579     \lngx_set_keys:n {
580       text~
581       italic~ features      = {
582         SizeFeatures        = {
583           {
584             Size              = {-8},
585             Font              = {
586               NewCM08-Italic.otf
587             }
588           },
589           {
590             Size              = {8-},
591             Font              = {
592               NewCM08-Italic.otf
593             }
594           }
595         }
596       }
597     }
598   }
599 }
600 \tl_if_in:cnT {
601   g _ lngx _ text _ italic _ tl
602 } { NewCMSans } {

```

```

603 \tl_if_in:cnTF {
604   g _ lngx _ text _ italic _ tl
605 } { Book } {
606   \lngx_set_keys:n {
607     text~
608     italic~ features      = {
609       SizeFeatures        = {
610         {
611           Size              = {-8},
612           Font              = {
613             NewCMSans08-BookOblique.otf
614           }
615         },
616         {
617           Size              = {8-},
618           Font              = {
619             NewCMSans10-BookOblique.otf
620           }
621         }
622       }
623     }
624   }
625 } {
626   \lngx_set_keys:n {
627     text~
628     italic~ features      = {
629       SizeFeatures        = {
630         {
631           Size              = {-8},
632           Font              = {
633             NewCMSans08-Oblique.otf
634           }
635         },
636         {
637           Size              = {8-},
638           Font              = {
639             NewCMSans08-Oblique.otf
640           }
641         }
642       }
643     }
644   }
645 }
646 }

```

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.

```

647 \tl_if_in:cnT {
648   g _ lngx _ text _ sans _ upright _ tl
649 } { NewCMSans } {
650   \tl_if_in:cnTF {
651     g _ lngx _ text _ upright _ tl
652   } { Book } {
653     \lngx_set_keys:n {

```



```

654     text~ sans~
655     upright~ features      = {
656         SizeFeatures      = {
657             {
658                 Size      = {-8},
659                 Font      = {
660                     NewCMSans08-Book.otf
661                 }
662             },
663             {
664                 Size      = {8-},
665                 Font      = {
666                     NewCMSans10-Book.otf
667                 }
668             }
669         }
670     }
671 }
672 } {
673     \lngx_set_keys:n {
674         text~ sans~
675         upright~ features      = {
676             SizeFeatures      = {
677                 {
678                     Size      = {-8},
679                     Font      = {
680                         NewCMSans08-Regular.otf
681                     }
682                 },
683                 {
684                     Size      = {8-},
685                     Font      = {
686                         NewCMSans10-Regular.otf
687                     }
688                 }
689             }
690         }
691     }
692 }
693 }
694 \tl_if_in:cnT {
695     g _ lngx _ text _ sans _ italic _ tl
696 } { NewCMSans } {
697     \tl_if_in:cnTF {
698         g _ lngx _ text _ italic _ tl
699     } { Book } {
700         \lngx_set_keys:n {
701             text~ sans~
702             italic~ features      = {
703                 SizeFeatures      = {
704                     {
705                         Size      = {-8},
706                         Font      = {
707                             NewCMSans08-BookOblique.otf

```

```

708         }
709     },
710     {
711         Size                = {8-},
712         Font                = {
713             NewCMSans10-BookOblique.otf
714         }
715     }
716 }
717 }
718 }
719 } {
720     \lngx_set_keys:n {
721         text~ sans~
722         italic~ features    = {
723             SizeFeatures    = {
724                 {
725                     Size      = {-8},
726                     Font      = {
727                         NewCMSans08-Oblique.otf
728                     }
729                 },
730                 {
731                     Size      = {8-},
732                     Font      = {
733                         NewCMSans10-Oblique.otf
734                     }
735                 }
736             }
737         }
738     }
739 }
740 }

```

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 LINGUISCIX feature keys.

```

741 \lngx_set_keys:n {
742     text~
743     upright~ features    = {
744         UprightFont      = {
745             \g_lngx_text_upright_tl
746         },
747         UprightFeatures   = {
748             \g_lngx_text_upright_features_tl
749         },
750         ItalicFont        = {
751             \g_lngx_text_italic_tl
752         },
753         ItalicFeatures    = {
754             \g_lngx_text_italic_features_tl
755         },
756         BoldFont          = {

```

```

757     \g_lngx_text_bold_upright_tl
758 },
759 BoldFeatures          = {
760     \g_lngx_text_bold_upright_features_tl
761 },
762 BoldItalicFont        = {
763     \g_lngx_text_bold_italic_tl
764 },
765 BoldItalicFeatures    = {
766     \g_lngx_text_bold_italic_features_tl
767 },

```

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).

```

768 \tl_if_empty:cF {
769     g _ lngx _ text _ slanted _ tl
770 } {
771     SlantedFont          = {
772         \g_lngx_text_slanted_tl
773     },
774     \tl_if_empty:cF {
775         g _ lngx _ text _ slanted _ features _ tl
776     } {
777         SlantedFeatures    = {
778             \g_lngx_text_slanted_features_tl
779         },
780     }
781 }
782 \tl_if_empty:cF {
783     g _ lngx _ text _ bold _ slanted _ tl
784 } {
785     BoldSlantedFont       = {
786         \g_lngx_text_bold_slanted_tl
787     },
788     BoldSlantedFeatures    = {
789         \g_lngx_text_bold_slanted_features_tl
790     },
791 }
792 \tl_if_empty:cF {
793     g _ lngx _ text _ swash _ tl
794 } {
795     SwashFont              = {
796         \g_lngx_text_swash_tl
797     },
798     SwashFeatures          = {
799         \g_lngx_text_swash_features_tl
800     },
801 }
802 \tl_if_empty:cF {
803     g _ lngx _ text _ bold _ swash _ tl
804 } {
805     BoldSwashFont          = {
806         \g_lngx_text_bold_swash_tl
807     },

```

```

808     BoldSwashFeatures      = {
809         \g_lngx_text_bold_swash_features_tl
810     },
811 }
812 \tl_if_empty:cF {
813     g _ lngx _ text _ small _ caps _ tl
814 } {
815     SmallCapsFont          = {
816         \g_lngx_text_small_caps_tl
817     },
818     SmallCapsFeatures      = {
819         \g_lngx_text_small_caps_features_tl
820     }
821 }
822 },

```

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

```

823 text~ sans~
824 upright~ features      = {
825     UprightFont          = {
826         \g_lngx_text_sans_upright_tl
827     },
828     UprightFeatures      = {
829         \g_lngx_text_sans_upright_features_tl
830     },
831     BoldFont              = {
832         \g_lngx_text_sans_bold_upright_tl
833     },
834     BoldFeatures          = {
835         \g_lngx_text_sans_bold_upright_features_tl
836     },
837     ItalicFont            = {
838         \g_lngx_text_sans_italic_tl
839     },
840     ItalicFeatures        = {
841         \g_lngx_text_sans_italic_features_tl
842     },
843     BoldItalicFont        = {
844         \g_lngx_text_sans_bold_italic_tl
845     },
846     BoldItalicFeatures    = {
847         \g_lngx_text_sans_bold_italic_features_tl
848     },
849     \tl_if_empty:cF {
850         g _ lngx _ text _ sans _ slanted _ tl
851     } {
852         SlantedFont        = {
853             \g_lngx_text_sans_slanted_tl
854         },
855         SlantedFeatures    = {
856             \g_lngx_text_sans_slanted_features_tl
857         },
858     }
859     \tl_if_empty:cF {
860         g _ lngx _ text _ sans _ bold _ slanted _ tl

```

```

861 } {
862     BoldSlantedFont      = {
863         \g_lngx_text_sans_bold_slanted_tl
864     },
865     BoldSlantedFeatures  = {
866         \g_lngx_text_sans_bold_slanted_features_tl
867     },
868 }
869 \tl_if_empty:cF {
870     g _ lngx _ text _ sans _ swash _ tl
871 } {
872     SwashFont            = {
873         \g_lngx_text_sans_swash_tl
874     },
875     SwashFeatures        = {
876         \g_lngx_text_sans_swash_features_tl
877     },
878 }
879 \tl_if_empty:cF {
880     g _ lngx _ text _ sans _ bold _ swash _ tl
881 } {
882     BoldSwashFont        = {
883         \g_lngx_text_sans_bold_swash_tl
884     },
885     BoldSwashFeatures    = {
886         \g_lngx_text_sans_bold_swash_features_tl
887     },
888 }
889 \tl_if_empty:cF {
890     g _ lngx _ text _ sans _ small _ caps _ tl
891 } {
892     SmallCapsFont        = {
893         \g_lngx_text_sans_small_caps_tl
894     },
895     SmallCapsFeatures    = {
896         \g_lngx_text_sans_small_caps_features_tl
897     }
898 }
899 },
900 text~ mono~
901 upright~ features      = {
902     UprightFont         = {
903         \g_lngx_text_mono_upright_tl
904     },
905     UprightFeatures      = {
906         \g_lngx_text_mono_upright_features_tl
907     },
908     BoldFont             = {
909         \g_lngx_text_mono_bold_upright_tl
910     },
911     BoldFeatures         = {
912         \g_lngx_text_mono_bold_upright_features_tl
913     },
914     ItalicFont           = {

```

```

915     \g_lngx_text_mono_italic_tl
916 },
917 ItalicFeatures      = {
918     \g_lngx_text_mono_italic_features_tl
919 },
920 BoldItalicFont      = {
921     \g_lngx_text_mono_bold_italic_tl
922 },
923 BoldItalicFeatures  = {
924     \g_lngx_text_mono_bold_italic_features_tl
925 },
926 \tl_if_empty:cF {
927     g _ lngx _ text _ mono _ slanted _ tl
928 } {
929     SlantedFont      = {
930         \g_lngx_text_mono_slanted_tl
931     },
932     SlantedFeatures  = {
933         \g_lngx_text_mono_slanted_features_tl
934     },
935 }
936 \tl_if_empty:cF {
937     g _ lngx _ text _ mono _ bold _ slanted _ tl
938 } {
939     BoldSlantedFont  = {
940         \g_lngx_text_mono_bold_slanted_tl
941     },
942     BoldSlantedFeatures = {
943         \g_lngx_text_mono_bold_slanted_features_tl
944     },
945 }
946 \tl_if_empty:cF {
947     g _ lngx _ text _ mono _ swash _ tl
948 } {
949     SwashFont        = {
950         \g_lngx_text_mono_swash_tl
951     },
952     SwashFeatures    = {
953         \g_lngx_text_mono_swash_features_tl
954     },
955 }
956 \tl_if_empty:cF {
957     g _ lngx _ text _ mono _ bold _ swash _ tl
958 } {
959     BoldSwashFont    = {
960         \g_lngx_text_mono_bold_swash_tl
961     },
962     BoldSwashFeatures = {
963         \g_lngx_text_mono_bold_swash_features_tl
964     },
965 }
966 \tl_if_empty:cF {
967     g _ lngx _ text _ mono _ small _ caps _ tl
968 } {

```

```

969     SmallCapsFont          = {
970         \g_lngx_text_mono_small_caps_tl
971     },
972     SmallCapsFeatures      = {
973         \g_lngx_text_mono_small_caps_features_tl
974     }
975 }
976 }
977 }
978 \bool_if:NT \g_lngx_old_style_bool {
979     \lngx_set_keys:n {
980         text~
981         upright~ features    = {
982             Numbers          = { OldStyle }
983         },
984         text~ sans~
985         upright~ features    = {
986             Numbers          = { OldStyle }
987         }
988     }
989     \tl_if_in:cnT {
990         g_lngx_math_tl
991     } { NewCM } {
992         \bool_if:NT \g_lngx_old_style_one_bool {
993             \lngx_set_keys:n {
994                 text~
995                 upright~ features    = {
996                     CharacterVariant = { 6 }
997                 },
998                 text~ sans~
999                 upright~ features    = {
1000                     CharacterVariant = { 6 }
1001                 }
1002             }
1003         }
1004     }
1005 }
1006 \tl_if_in:cnT {
1007     g _ lngx _ math _ tl
1008 } { NewCM } {
1009     \bool_if:NT \g_lngx_bourbaki_bool {
1010         \lngx_set_keys:n {
1011             math~ features    = {
1012                 CharacterVariant = { 1 }
1013             }
1014         }
1015     }
1016 }

```

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.

```

1017 \tl_if_in:cnT {
1018     g _ lngx _ text _ upright _ tl

```

```

1019 } { NewCM } {
1020   \lngx_set_keys:n {
1021     text~
1022     upright~ features          = {
1023       IgnoreFontspecFile
1024     }
1025   }
1026 }
1027 \tl_if_in:cnT {
1028   g _ lngx _ text _ sans _ upright _ tl
1029 } { NewCM } {
1030   \lngx_set_keys:n {
1031     text~
1032     sans~ upright~ features    = {
1033       IgnoreFontspecFile
1034     }
1035   }
1036 }
1037 \tl_if_in:cnT {
1038   g _ lngx _ text _ mono _ upright _ tl
1039 } { NewCM } {
1040   \lngx_set_keys:n {
1041     text~
1042     mono~ upright~ features    = {
1043       IgnoreFontspecFile
1044     }
1045   }
1046 }
1047 \lngx_set_main_font:ee {
1048   \g_lngx_text_upright_features_tl
1049 } {
1050   \g_lngx_text_upright_tl
1051 }
1052 \lngx_set_sans_font:ee {
1053   \g_lngx_text_sans_upright_features_tl
1054 } {
1055   \g_lngx_text_sans_upright_tl
1056 }
1057 \lngx_set_mono_font:ee {
1058   \g_lngx_text_mono_upright_features_tl
1059 } {
1060   \g_lngx_text_mono_upright_tl
1061 }
1062 \lngx_set_math_font:ee {
1063   \g_lngx_math_features_tl
1064 } {
1065   \g_lngx_math_tl
1066 }
1067 }
1068 </font>

```



```

1069 <*ipa>
1070 \ProvidesExplPackage{linguistix-ipa}
1071     {2025-05-29}
1072     {v0.2}
1073     {%
1074     A package for typesetting the IPA
1075     (International Phonetic Alphabet) from
1076     the ‘LinguisTiX’ bundle.%
1077     }

```

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

```

1078
1079 \IfPackageLoadedF { unicode-math } {
1080   \RequirePackage { unicode-math }
1081 }
1082
1083 \IfPackageLoadedF { linguistix-base } {
1084   \RequirePackage { linguistix-base }
1085 }
1086
1087 \IfPackageLoadedF { linguistix-nfss } {
1088   \RequirePackage { linguistix-nfss }
1089 }

```

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

```

1090
1091 \NewDocumentCommand \ipatext { s m } {
1092   \IfBooleanTF { #1 } {
1093     {
1094       \lngxipa
1095       / #2 /
1096     }
1097   } {
1098     {
1099       \lngxipa
1100       [ #2 ]
1101     }
1102   }
1103 }

```

(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

```

```

II04
II05 \clist_map_inline:nn {
II06   upright,
II07   bold~ upright,
II08   italic,
II09   bold~ italic,
II10   slanted,
II11   bold~ slanted,
II12   swash,
II13   bold~ swash,
II14   small~ caps
II15 } {
II16   \tl_set:Nn \l_tmpa_tl { #1 }
II17   \tl_replace_all:Nnn \l_tmpa_tl { ~ } { _ }
II18   \tl_gclear_new:c {
II19     g _ lngx _ ipa _ \l_tmpa_tl _ features _ tl
II20   }
II21   \keys_define:nn { lngx _ keys } {
II22     ipa~ #1
II23     .tl_gset_e:c          = {
II24       g _ lngx _ ipa _ \l_tmpa_tl _ tl
II25     },
II26     ipa~ #1~ features
II27     .code:n              = {
II28       \tl_set:Nn \l_tmpb_tl { #1 }
II29       \tl_replace_all:Nnn \l_tmpb_tl { ~ } { _ }
II30       \tl_put_right:ce {
II31         g _ lngx _ ipa _ \l_tmpb_tl _ features _ tl
II32       } { ##1 , }
II33       \tl_clear:N \l_tmpb_tl
II34     }
II35   }
II36   \tl_clear:N \l_tmpa_tl
II37 }

```

(End of definition for *ipa upright* and others. These functions are documented on page 7.)

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

```

```

1138
1139 \clist_map_inline:nn {
1140   upright,
1141   bold~ upright,
1142   italic,
1143   bold~ italic,
1144   slanted,
1145   bold~ slanted,
1146   swash,
1147   bold~ swash,
1148   small~ caps
1149 } {
1150   \tl_set:Nn \l_tmpa_tl { #1 }
1151   \tl_replace_all:Nnn \l_tmpa_tl { ~ } { _ }
1152   \tl_gclear_new:c {
1153     g _ lngx _ ipa _ mono _ \l_tmpa_tl _ features _ tl
1154   }
1155   \keys_define:nn { lngx _ keys } {
1156     ipa~ mono~ #1
1157     .tl_gset_e:c          = {
1158       g _ lngx _ ipa _ mono _ \l_tmpa_tl _ tl
1159     },
1160     ipa~ mono~ #1~ features
1161     .code:n              = {
1162       \tl_set:Nn \l_tmpb_tl { #1 }
1163       \tl_replace_all:Nnn \l_tmpb_tl { ~ } { _ }
1164       \tl_put_right:ce {
1165         g _ lngx _ ipa _ mono _ \l_tmpb_tl _ features _ tl
1166       } { ##1 , }
1167       \tl_clear:N \l_tmpb_tl
1168     }
1169   }
1170   \tl_clear:N \l_tmpa_tl
1171 }

```

(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 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
ipa newcm slanted
ipa newcm slanted features
ipa newcm swash
ipa newcm swash features
ipa newcm small caps
ipa newcm small caps 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 = {

```

```

I220     NewCM10-BookItalic.otf
I221 },
I222 ipa~
I223 bold~ italic          = {
I224     NewCM10-BoldItalic.otf
I225 },
I226 ipa~
I227 slanted               = {
I228     NewCM10-Book.otf
I229 },
I230 ipa~
I231 bold~ slanted         = {
I232     NewCM10-Bold.otf
I233 },
I234 ipa~
I235 swash                 = {
I236     NewCM10-Book.otf
I237 },
I238 ipa~
I239 bold~ swash           = {
I240     NewCM10-Bold.otf
I241 },
I242 ipa~
I243 small~ caps           = {
I244     NewCM10-Book.otf
I245 },
I246 ipa~
I247 sans~ upright        = {
I248     NewCMSans10-Book.otf
I249 },
I250 ipa~
I251 sans~ bold~ upright   = {
I252     NewCMSans10-Bold.otf
I253 },
I254 ipa~
I255 sans~ italic          = {
I256     NewCMSans10-BookOblique.otf
I257 },
I258 ipa~
I259 sans~ bold~ italic    = {
I260     NewCMSans10-BoldOblique.otf
I261 },
I262 ipa~
I263 sans~ slanted         = {
I264     NewCMSans10-BookOblique.otf
I265 },
I266 ipa~
I267 sans~ bold~ slanted   = {
I268     NewCMSans10-BoldOblique.otf
I269 },
I270 ipa~
I271 sans~ swash           = {
I272     NewCMSans10-Book.otf
I273 },

```

```

1274 ipa~
1275 sans~ bold~ swash      = {
1276     NewCMSans10-Bold.otf
1277 },
1278 ipa~
1279 sans~ small~ caps      = {
1280     NewCMSans10-Book.otf
1281 },
1282 ipa~
1283 mono~ upright          = {
1284     NewCMMono10-Book.otf
1285 },
1286 ipa~
1287 mono~ bold~ upright    = {
1288     NewCMMono10-Bold.otf
1289 },
1290 ipa~
1291 mono~ italic           = {
1292     NewCMMono10-BookItalic.otf
1293 },
1294 ipa~
1295 mono~ bold~ italic     = {
1296     NewCMMono10-BoldOblique.otf
1297 },
1298 ipa~
1299 mono~ slanted          = {
1300     NewCMMono10-Book.otf
1301 },
1302 ipa~
1303 mono~ bold~ slanted    = {
1304     NewCMMono10-BoldOblique.otf
1305 },
1306 ipa~
1307 mono~ swash            = {
1308     NewCMMono10-Book.otf
1309 },
1310 ipa~
1311 mono~ bold~ swash      = {
1312     NewCMMono10-Bold.otf
1313 },
1314 ipa~
1315 mono~ small~ caps      = {
1316     NewCMMono10-Book.otf
1317 }
1318 }
1319 }

```

(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.

```

1320
1321 \keys_define:nn { lngx _ keys } {
1322   ipa~ newcm~ sans

```

```

1323 .meta:n          = {
1324   ipa~
1325   upright          = {
1326     NewCMSans10-Book.otf
1327   },
1328   ipa~
1329   bold~ upright    = {
1330     NewCMSans10-Bold.otf
1331   },
1332   ipa~
1333   italic           = {
1334     NewCMSans10-BookOblique.otf
1335   },
1336   ipa~
1337   bold~ italic     = {
1338     NewCMSans10-BoldOblique.otf
1339   },
1340   ipa~
1341   slanted          = {
1342     NewCMSans10-BookOblique.otf
1343   },
1344   ipa~
1345   bold~ slanted    = {
1346     NewCMSans10-BoldOblique.otf
1347   },
1348   ipa~
1349   swash            = {
1350     NewCMSans10-Book.otf
1351   },
1352   ipa~
1353   bold~ swash      = {
1354     NewCMSans10-Bold.otf
1355   },
1356   ipa~
1357   small~ caps      = {
1358     NewCMSans10-Book.otf
1359   }
1360 }
1361 }

```

(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.

```

1362
1363 \keys_define:nn { lngx _ keys } {
1364   ipa~ newcm~ mono
1365   .meta:n          = {
1366     ipa~
1367     upright          = {
1368       NewCMMono10-Book.otf
1369     },
1370     ipa~
1371     bold~ upright    = {

```

```

1372     NewCMMono10-Bold.otf
1373 },
1374 ipa~
1375 italic                = {
1376     NewCMMono10-BookItalic.otf
1377 },
1378 ipa~
1379 bold~ italic          = {
1380     NewCMMono10-BoldOblique.otf
1381 },
1382 ipa~
1383 slanted                = {
1384     NewCMMono10-Book.otf
1385 },
1386 ipa~
1387 bold~ slanted          = {
1388     NewCMMono10-BoldOblique.otf
1389 },
1390 ipa~
1391 swash                  = {
1392     NewCMMono10-Book.otf
1393 },
1394 ipa~
1395 bold~ swash            = {
1396     NewCMMono10-Bold.otf
1397 },
1398 ipa~
1399 small~ caps            = {
1400     NewCMMono10-Book.otf
1401 }
1402 }
1403 }

```

(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.

```

1404
1405 \keys_define:nn { lngx _ keys } {
1406   ipa~ newcm~ regular
1407   .meta:n                = {
1408     ipa~
1409     upright                = {
1410         NewCM10-Regular.otf
1411     },
1412     ipa~
1413     bold~ upright          = {
1414         NewCM10-Bold.otf
1415     },
1416     ipa~
1417     italic                 = {
1418         NewCM10-Italic.otf
1419     },
1420     ipa~

```



```

I421 bold~ italic = {
I422     NewCM10-BoldItalic.otf
I423 },
I424 ipa~
I425 slanted = {
I426     NewCM10-Regular.otf
I427 },
I428 ipa~
I429 bold~ slanted = {
I430     NewCM10-Bold.otf
I431 },
I432 ipa~
I433 swash = {
I434     NewCM10-Regular.otf
I435 },
I436 ipa~
I437 bold~ swash = {
I438     NewCM10-Bold.otf
I439 },
I440 ipa~
I441 small~ caps = {
I442     NewCM10-Regular.otf
I443 },
I444 ipa~
I445 sans~ upright = {
I446     NewCMSans10-Regular.otf
I447 },
I448 ipa~
I449 sans~ bold = {
I450     NewCMSans10-Bold.otf
I451 },
I452 ipa~
I453 sans~ italic = {
I454     NewCMSans10-Oblique.otf
I455 },
I456 ipa~
I457 sans~ bold~ italic = {
I458     NewCMSans10-BoldOblique.otf
I459 },
I460 ipa~
I461 sans~ slanted = {
I462     NewCMSans10-Regular.otf
I463 },
I464 ipa~
I465 sans~ bold~ slanted = {
I466     NewCMSans10-Bold.otf
I467 },
I468 ipa~
I469 sans~ swash = {
I470     NewCMSans10-Regular.otf
I471 },
I472 ipa~
I473 sans~ bold~ swash = {
I474     NewCMSans10-Bold.otf

```

```

1475 },
1476 ipa~
1477 sans~ small~ caps      = {
1478   NewCMSans10-Regular.otf
1479 },
1480 ipa~
1481 mono~ upright          = {
1482   NewCMMono10-Regular.otf
1483 },
1484 ipa~
1485 mono~ bold             = {
1486   NewCMMono10-Bold.otf
1487 },
1488 ipa~
1489 mono~ italic           = {
1490   NewCMMono10-Italic.otf
1491 },
1492 ipa~
1493 mono~ bold~ italic     = {
1494   NewCMMono10-Bold.otf
1495 },
1496 ipa~
1497 mono~ slanted          = {
1498   NewCMMono10-Regular.otf
1499 },
1500 ipa~
1501 mono~ bold~ slanted    = {
1502   NewCMMono10-Bold.otf
1503 },
1504 ipa~
1505 mono~ swash            = {
1506   NewCMMono10-Regular.otf
1507 },
1508 ipa~
1509 mono~ bold~ swash      = {
1510   NewCMMono10-Bold.otf
1511 },
1512 ipa~
1513 mono~ small~ caps      = {
1514   NewCMMono10-Regular.otf
1515 }
1516 }
1517 }

```

(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.

```

1518
1519 \keys_define:nn { lngx _ keys } {
1520   ipa~ newcm~ sans~ regular
1521   .meta:n          = {
1522     ipa~
1523     upright          = {

```

```

1524     NewCMSans10-Regular.otf
1525 },
1526 ipa~
1527 bold                      = {
1528     NewCMSans10-Bold.otf
1529 },
1530 ipa~
1531 italic                    = {
1532     NewCMSans10-Oblique.otf
1533 },
1534 ipa~
1535 bold~ italic              = {
1536     NewCMSans10-BoldOblique.otf
1537 },
1538 ipa~
1539 slanted                   = {
1540     NewCMSans10-Regular.otf
1541 },
1542 ipa~
1543 bold~ slanted             = {
1544     NewCMSans10-Bold.otf
1545 },
1546 ipa~
1547 swash                     = {
1548     NewCMSans10-Regular.otf
1549 },
1550 ipa~
1551 bold~ swash               = {
1552     NewCMSans10-Bold.otf
1553 },
1554 ipa~
1555 small~ caps               = {
1556     NewCMSans10-Regular.otf
1557 }
1558 }
1559 }

```

(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.

```

1560
1561 \keys_define:nn { lngx _ keys } {
1562   ipa~ newcm~ mono~ regular
1563   .meta:n                = {
1564     ipa~
1565     upright                = {
1566         NewCMMono10-Regular.otf
1567     },
1568     ipa~
1569     bold                    = {
1570         NewCMMono10-Bold.otf
1571     },
1572     ipa~

```

```

1573     italic                = {
1574       NewCMMono10-Italic.otf
1575     },
1576     ipa~
1577     bold~ italic          = {
1578       NewCMMono10-Bold.otf
1579     },
1580     ipa~
1581     slanted                = {
1582       NewCMMono10-Regular.otf
1583     },
1584     ipa~
1585     bold~ slanted          = {
1586       NewCMMono10-Bold.otf
1587     },
1588     ipa~
1589     swash                  = {
1590       NewCMMono10-Regular.otf
1591     },
1592     ipa~
1593     bold~ swash            = {
1594       NewCMMono10-Bold.otf
1595     },
1596     ipa~
1597     small~ caps            = {
1598       NewCMMono10-Regular.otf
1599     }
1600   }
1601 }

```

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

We set the `ipa newcm` key by default.

```

1602
1603 \ltx_set_keys:n {ipa~ newcm}

```

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

```

1604
1605 \sys_if_engine luatex:T {
1606   \ltx_set_keys:n {
1607     ipa~
1608     upright~ features      = {
1609       Renderer              = { HarfBuzz }
1610     },
1611     ipa~ sans~
1612     upright~ features      = {
1613       Renderer              = { HarfBuzz }
1614     },
1615     ipa~ mono~
1616     upright~ features      = {
1617       Renderer              = { HarfBuzz }
1618     }
1619   }
1620 }

```

`\lngx_set_main_ipa_font:nn`
`\lngx_main_ipa:`
`lngx_ipa_rm_nfss`
`\lngx_set_sans_ipa_font:nn`
`\lngx_sans_ipa:`
`lngx_ipa_sf_nfss`
`\lngx_set_mono_ipa_font:nn`
`\lngx_mono_ipa:`
`lngx_ipa_tt_nfss`

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

```

r621
r622 \cs_new_protected:Npn \lngx_set_main_ipa_font:nn #1#2 {
r623   \setfontfamily \lngx_main_ipa: [
r624     #1,
r625     NFSSFamily           = { lngx _ ipa _ rm _ nfss }
r626   ] { #2 }
r627 }
r628
r629 \cs_new_protected:Npn \lngx_set_sans_ipa_font:nn #1#2 {
r630   \setfontfamily \lngx_sans_ipa: [
r631     #1,
r632     NFSSFamily           = { lngx _ ipa _ sf _ nfss }
r633   ] { #2 }
r634 }
r635
r636 \cs_new_protected:Npn \lngx_set_mono_ipa_font:nn #1#2 {
r637   \setfontfamily \lngx_mono_ipa: [
r638     #1,
r639     NFSSFamily           = { lngx _ ipa _ tt _ nfss }
r640   ] { #2 }
r641 }
r642
r643 \cs_generate_variant:Nn \lngx_set_main_ipa_font:nn { ee }
r644 \cs_generate_variant:Nn \lngx_set_sans_ipa_font:nn { ee }
r645 \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 LINGUISTICX-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.

```

r646
r647 \lngx_super_font_family:nn { lngx _ ipa } {
r648   rm           = { lngx _ ipa _ rm _ nfss },
r649   sf           = { lngx _ ipa _ sf _ nfss },
r650   tt           = { lngx _ ipa _ tt _ nfss }
r651 }

```

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

\lngxipa
\lngx_ipa:

I use `\lngx_softer_super_font_family:n` provided by LINGUISTICX-NFSS for defining this switch to the IPA.

```

r652
r653 \cs_new_protected:Npn \lngx_ipa: {
r654   \lngx_softer_super_font_family:n { lngx _ ipa }
r655 }
r656
r657 \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 `LINGUISCIX-FONTS` for setting the size variants. This is done with lazy evaluation, just before `\begin{document}`.

```

1658
1659 \hook_gput_code:nnn { begindocument / before } { . } {
1660   \tl_if_in:cnT {
1661     g _ lngx _ ipa _ upright _ tl
1662   } { NewCM } {
1663     \tl_if_in:cnTF {
1664       g _ lngx _ ipa _ upright _ tl
1665     } { Book } {
1666       \lngx_set_keys:n {
1667         ipa~
1668         upright~ features      = {
1669           SizeFeatures         = {
1670             {
1671               Size              = {-8},
1672               Font              = {
1673                 NewCM08-Book.otf
1674             }
1675           },
1676           {
1677             Size                = {8-},
1678             Font                = {
1679               NewCM10-Book.otf
1680             }
1681           }
1682         }
1683       }
1684     }
1685   } {
1686     \lngx_set_keys:n {
1687       ipa~
1688       upright~ features      = {
1689         SizeFeatures         = {
1690           {
1691             Size                = {-8},
1692             Font                = {
1693               NewCM08-Regular.otf
1694             }
1695           },
1696           {
1697             Size                = {8-},
1698             Font                = {
1699               NewCM10-Regular.otf
1700             }
1701           }
1702         }
1703       }
1704     }
1705   }
1706 }
1707 \tl_if_in:cnT {

```

```

1708     g _ lngx _ ipa _ upright _ tl
1709 } { NewCMSans } {
1710     \tl_if_in:cnTF {
1711         g _ lngx _ ipa _ upright _ tl
1712     } { Book } {
1713         \lngx_set_keys:n {
1714             ipa~
1715             upright~ features      = {
1716                 SizeFeatures      = {
1717                     {
1718                         Size        = {-8},
1719                         Font        = {
1720                             NewCMSans08-Book.otf
1721                         }
1722                     },
1723                     {
1724                         Size        = {8-},
1725                         Font        = {
1726                             NewCMSans10-Book.otf
1727                         }
1728                     }
1729                 }
1730             }
1731         }
1732     } {
1733         \lngx_set_keys:n {
1734             ipa~
1735             upright~ features      = {
1736                 SizeFeatures      = {
1737                     {
1738                         Size        = {-8},
1739                         Font        = {
1740                             NewCMSans08-Regular.otf
1741                         }
1742                     },
1743                     {
1744                         Size        = {8-},
1745                         Font        = {
1746                             NewCMSans10-Regular.otf
1747                         }
1748                     }
1749                 }
1750             }
1751         }
1752     }
1753 }
1754 \tl_if_in:cnT {
1755     g _ lngx _ ipa _ italic _ tl
1756 } { NewCM } {
1757     \tl_if_in:cnTF {
1758         g _ lngx _ ipa _ italic _ tl
1759     } { Book } {
1760         \lngx_set_keys:n {
1761             ipa~

```

```

1762         italic~ features          = {
1763             SizeFeatures           = {
1764                 {
1765                     Size             = {-8},
1766                     Font             = {
1767                         NewCM08-BookItalic.otf
1768                     }
1769                 },
1770                 {
1771                     Size             = {8-},
1772                     Font             = {
1773                         NewCM10-BookItalic.otf
1774                     }
1775                 }
1776             }
1777         }
1778     }
1779 } {
1780     \lngx_set_keys:n {
1781         ipa~
1782         italic~ features          = {
1783             SizeFeatures           = {
1784                 {
1785                     Size             = {-8},
1786                     Font             = {
1787                         NewCM08-Italic.otf
1788                     }
1789                 },
1790                 {
1791                     Size             = {8-},
1792                     Font             = {
1793                         NewCM08-Italic.otf
1794                     }
1795                 }
1796             }
1797         }
1798     }
1799 }
1800 }
1801 \tl_if_in:cnT {
1802     g _ lngx _ ipa _ italic _ tl
1803 } { NewCMSans } {
1804     \tl_if_in:cnTF {
1805         g _ lngx _ ipa _ italic _ tl
1806     } { Book } {
1807         \lngx_set_keys:n {
1808             ipa~
1809             italic~ features          = {
1810                 SizeFeatures           = {
1811                     {
1812                         Size             = {-8},
1813                         Font             = {
1814                             NewCMSans08-BookOblique.otf
1815                     }

```



```

1816         },
1817         {
1818             Size              = {8-},
1819             Font              = {
1820                 NewCMSans10-BookOblique.otf
1821             }
1822         }
1823     }
1824 }
1825 }
1826 } {
1827     \lngx_set_keys:n {
1828         ipa~
1829         italic~ features      = {
1830             SizeFeatures      = {
1831                 {
1832                     Size              = {-8},
1833                     Font              = {
1834                         NewCMSans08-Oblique.otf
1835                     }
1836                 },
1837                 {
1838                     Size              = {8-},
1839                     Font              = {
1840                         NewCMSans08-Oblique.otf
1841                     }
1842                 }
1843             }
1844         }
1845     }
1846 }
1847 }
1848 \tl_if_in:cnT {
1849     g _ lngx _ ipa _ sans _ upright _ tl
1850 } { NewCMSans } {
1851     \tl_if_in:cnTF {
1852         g _ lngx _ ipa _ upright _ tl
1853     } { Book } {
1854         \lngx_set_keys:n {
1855             ipa~ sans~
1856             upright~ features      = {
1857                 SizeFeatures      = {
1858                     {
1859                         Size              = {-8},
1860                         Font              = {
1861                             NewCMSans08-Book.otf
1862                         }
1863                     },
1864                     {
1865                         Size              = {8-},
1866                         Font              = {
1867                             NewCMSans10-Book.otf
1868                         }
1869                     }

```

```

1870     }
1871   }
1872 }
1873 } {
1874   \lngx_set_keys:n {
1875     ipa~ sans~
1876     upright~ features      = {
1877       SizeFeatures        = {
1878         {
1879           Size             = {-8},
1880           Font             = {
1881             NewCMSans08-Regular.otf
1882           }
1883         },
1884         {
1885           Size             = {8-},
1886           Font             = {
1887             NewCMSans10-Regular.otf
1888           }
1889         }
1890       }
1891     }
1892   }
1893 }
1894 }
1895 \tl_if_in:cnT {
1896   g _ lngx _ ipa _ sans _ italic _ tl
1897 } { NewCMSans } {
1898   \tl_if_in:cnTF {
1899     g _ lngx _ ipa _ italic _ tl
1900   } { Book } {
1901     \lngx_set_keys:n {
1902       ipa~ sans~
1903       italic~ features      = {
1904         SizeFeatures        = {
1905           {
1906             Size           = {-8},
1907             Font           = {
1908               NewCMSans08-BookOblique.otf
1909             }
1910           },
1911           {
1912             Size           = {8-},
1913             Font           = {
1914               NewCMSans10-BookOblique.otf
1915             }
1916           }
1917         }
1918       }
1919     }
1920   } {
1921     \lngx_set_keys:n {
1922       ipa~ sans~
1923       italic~ features      = {

```

```

1924         SizeFeatures          = {
1925             {
1926                 Size            = {-8},
1927                 Font            = {
1928                     NewCMSans08-Oblique.otf
1929                 }
1930             },
1931             {
1932                 Size            = {8-},
1933                 Font            = {
1934                     NewCMSans10-Oblique.otf
1935                 }
1936             }
1937         }
1938     }
1939 }
1940 }
1941 }

```

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

```

1942 \lngx_set_keys:n {
1943     ipa~ upright~ features = {
1944         UprightFont        = {
1945             \g_lngx_ipa_upright_tl
1946         },
1947         UprightFeatures    = {
1948             \g_lngx_ipa_upright_features_tl
1949         },
1950         BoldFont           = {
1951             \g_lngx_ipa_bold_upright_tl
1952         },
1953         BoldFeatures       = {
1954             \g_lngx_ipa_bold_upright_features_tl
1955         },
1956         ItalicFont         = {
1957             \g_lngx_ipa_italic_tl
1958         },
1959         ItalicFeatures     = {
1960             \g_lngx_ipa_italic_features_tl
1961         },
1962         BoldItalicFont     = {
1963             \g_lngx_ipa_bold_italic_tl
1964         },
1965         BoldItalicFeatures = {
1966             \g_lngx_ipa_bold_italic_features_tl
1967         },
1968         \tl_if_empty:cF {
1969             g _ lngx _ ipa _ slanted _ tl
1970         } {
1971             SlantedFont     = {
1972                 \g_lngx_ipa_slanted_tl
1973             },
1974             \tl_if_empty:cF {
1975                 g _ lngx _ ipa _ slanted _ features _ tl
1976             } {

```

```

1977         SlantedFeatures           = {
1978             \g_lngx_ipa_slanted_features_tl
1979         },
1980     }
1981 }
1982 \tl_if_empty:cF {
1983     g _ lngx _ ipa _ bold _ slanted _ tl
1984 } {
1985     BoldSlantedFont           = {
1986         \g_lngx_ipa_bold_slanted_tl
1987     },
1988     \tl_if_empty:cF {
1989         g _ lngx _ ipa _ bold _ slanted _ features _ tl
1990     } {
1991         BoldSlantedFeatures    = {
1992             \g_lngx_ipa_bold_slanted_features_tl
1993         },
1994     }
1995 }
1996 \tl_if_empty:cF {
1997     g _ lngx _ ipa _ swash _ tl
1998 } {
1999     SwashFont                 = {
2000         \g_lngx_ipa_swash_tl
2001     },
2002     \tl_if_empty:cF {
2003         g _ lngx _ ipa _ swash _ features _ tl
2004     } {
2005         SwashFeatures          = {
2006             \g_lngx_ipa_swash_features_tl
2007         },
2008     }
2009 }
2010 \tl_if_empty:cF {
2011     g _ lngx _ ipa _ bold _ swash _ tl
2012 } {
2013     BoldSwashFont             = {
2014         \g_lngx_ipa_bold_swash_tl
2015     },
2016     \tl_if_empty:cF {
2017         g _ lngx _ ipa _ bold _ swash _ features _ tl
2018     } {
2019         BoldSwashFeatures      = {
2020             \g_lngx_ipa_bold_swash_features_tl
2021         },
2022     }
2023 }
2024 \tl_if_empty:cF {
2025     g _ lngx _ ipa _ small _ caps _ tl
2026 } {
2027     SmallCapsFont             = {
2028         \g_lngx_ipa_small_caps_tl
2029     }
2030     \tl_if_empty:cF {

```

```

2031         g _ lngx _ ipa _ small _ caps _ features _ tl
2032     } {
2033         SmallCapsFeatures          = {
2034             \g_lngx_ipa_small_caps_features_tl
2035         }
2036     }
2037 }
2038 },
2039 ipa~
2040 sans~ upright~ features = {
2041     UprightFont          = {
2042         \g_lngx_ipa_sans_upright_tl
2043     },
2044     UprightFeatures      = {
2045         \g_lngx_ipa_sans_upright_features_tl
2046     },
2047     BoldFont             = {
2048         \g_lngx_ipa_sans_bold_upright_tl
2049     },
2050     BoldFeatures         = {
2051         \g_lngx_ipa_sans_bold_upright_features_tl
2052     },
2053     ItalicFont           = {
2054         \g_lngx_ipa_sans_italic_tl
2055     },
2056     ItalicFeatures       = {
2057         \g_lngx_ipa_sans_italic_features_tl
2058     },
2059     BoldItalicFont       = {
2060         \g_lngx_ipa_sans_bold_italic_tl
2061     },
2062     BoldItalicFeatures   = {
2063         \g_lngx_ipa_sans_bold_italic_features_tl
2064     },
2065     \tl_if_empty:cF {
2066         g _ lngx _ ipa _ slanted _ tl
2067     } {
2068         SlantedFont       = {
2069             \g_lngx_ipa_slanted_tl
2070         },
2071         \tl_if_empty:cF {
2072             g _ lngx _ ipa _ slanted _ features _ tl
2073         } {
2074             SlantedFeatures = {
2075                 \g_lngx_ipa_slanted_features_tl
2076             },
2077         }
2078     }
2079     \tl_if_empty:cF {
2080         g _ lngx _ ipa _ sans _ bold _ slanted _ tl
2081     } {
2082         BoldSlantedFont   = {
2083             \g_lngx_ipa_sans_bold_slanted_tl
2084         },

```

```

2085 \tl_if_empty:cF {
2086   g _ lngx _ ipa _ sans _ bold _ slanted _ features
2087   _ tl
2088 } {
2089   BoldSlantedFeatures = {
2090     \g_lngx_ipa_sans_bold_slanted_features_tl
2091   },
2092 }
2093 }
2094 \tl_if_empty:cF {
2095   g _ lngx _ ipa _ sans _ swash _ tl
2096 } {
2097   SwashFont = {
2098     \g_lngx_ipa_sans_swash_tl
2099   },
2100   \tl_if_empty:cF {
2101     g _ lngx _ ipa _ sans _ swash _ features _ tl
2102   } {
2103     SwashFeatures = {
2104       \g_lngx_ipa_sans_swash_features_tl
2105     },
2106   }
2107 }
2108 \tl_if_empty:cF {
2109   g _ lngx _ ipa _ sans _ bold _ swash _ tl
2110 } {
2111   BoldSwashFont = {
2112     \g_lngx_ipa_sans_bold_swash_tl
2113   },
2114   \tl_if_empty:cF {
2115     g _ lngx _ ipa _ sans _ bold _ swash _ features _
2116     tl
2117   } {
2118     BoldSwashFeatures = {
2119       \g_lngx_ipa_sans_bold_swash_features_tl
2120     },
2121   }
2122 }
2123 \tl_if_empty:cF {
2124   g _ lngx _ ipa _ sans _ small _ caps _ tl
2125 } {
2126   SmallCapsFont = {
2127     \g_lngx_ipa_sans_small_caps_tl
2128   }
2129   \tl_if_empty:cF {
2130     g _ lngx _ ipa _ sans _ small _ caps _ features _
2131     tl
2132   } {
2133     SmallCapsFeatures = {
2134       \g_lngx_ipa_sans_small_caps_features_tl
2135     }
2136   }
2137 }
2138 },

```

```

2139 ipa~
2140 mono~ upright~ features = {
2141     UprightFont = {
2142         \g_lngx_ipa_mono_bold_upright_tl
2143     },
2144     UprightFeatures = {
2145         \g_lngx_ipa_mono_bold_upright_features_tl
2146     },
2147     BoldFont = {
2148         \g_lngx_ipa_mono_bold_upright_tl
2149     },
2150     BoldFeatures = {
2151         \g_lngx_ipa_mono_bold_upright_features_tl
2152     },
2153     ItalicFont = {
2154         \g_lngx_ipa_mono_italic_tl
2155     },
2156     ItalicFeatures = {
2157         \g_lngx_ipa_mono_italic_features_tl
2158     },
2159     BoldItalicFont = {
2160         \g_lngx_ipa_mono_bold_italic_tl
2161     },
2162     BoldItalicFeatures = {
2163         \g_lngx_ipa_mono_bold_italic_features_tl
2164     },
2165     \tl_if_empty:cF {
2166         g _ lngx _ ipa _ mono _ slanted _ tl
2167     } {
2168         SlantedFont = {
2169             \g_lngx_ipa_mono_slanted_tl
2170         },
2171         \tl_if_empty:cF {
2172             g _ lngx _ ipa _ mono _ slanted _ features _ tl
2173         } {
2174             SlantedFeatures = {
2175                 \g_lngx_ipa_mono_slanted_features_tl
2176             },
2177         }
2178     }
2179     \tl_if_empty:cF {
2180         g _ lngx _ ipa _ mono _ bold _ slanted _ tl
2181     } {
2182         BoldSlantedFont = {
2183             \g_lngx_ipa_mono_bold_slanted_tl
2184         },
2185         \tl_if_empty:cF {
2186             g _ lngx _ ipa _ mono _ bold _ slanted _ features
2187             _ tl
2188         } {
2189             BoldSlantedFeatures = {
2190                 \g_lngx_ipa_mono_bold_slanted_features_tl
2191             },
2192         }

```

```

2193     }
2194     \tl_if_empty:cF {
2195         g _ lngx _ ipa _ mono _ swash _ tl
2196     } {
2197         SwashFont = {
2198             \g_lngx_ipa_mono_swash_tl
2199         },
2200         \tl_if_empty:cF {
2201             g _ lngx _ ipa _ mono _ swash _ features _ tl
2202         } {
2203             SwashFeatures = {
2204                 \g_lngx_ipa_mono_swash_features_tl
2205             },
2206         }
2207     }
2208     \tl_if_empty:cF {
2209         g _ lngx _ ipa _ mono _ bold _ swash _ tl
2210     } {
2211         BoldSwashFont = {
2212             \g_lngx_ipa_mono_bold_swash_tl
2213         },
2214         \tl_if_empty:cF {
2215             g _ lngx _ ipa _ mono _ bold _ swash _ features _
2216             tl
2217         } {
2218             BoldSwashFeatures = {
2219                 \g_lngx_ipa_mono_bold_swash_features_tl
2220             },
2221         }
2222     }
2223     \tl_if_empty:cF {
2224         g _ lngx _ ipa _ mono _ small _ caps _ tl
2225     } {
2226         SmallCapsFont = {
2227             \g_lngx_ipa_mono_small_caps_tl
2228         }
2229         \tl_if_empty:cF {
2230             g _ lngx _ ipa _ mono _ small _ caps _ features _
2231             tl
2232         } {
2233             SmallCapsFeatures = {
2234                 \g_lngx_ipa_mono_small_caps_features_tl
2235             }
2236         }
2237     }
2238 }
2239 }
2240 \tl_if_in:cnT {
2241     g _ lngx _ ipa _ upright _ tl
2242 } { NewCM } {
2243     \lngx_set_keys:n {
2244         ipa~
2245         upright~ features = {
2246             IgnoreFontspecFile,

```



```

2247         StylisticSet           = { 5 }
2248     }
2249 }
2250
2251 \tl_if_in:cnT {
2252   g _ lngx _ ipa _ sans _ upright _ tl
2253 } { NewCM } {
2254   \lngx_set_keys:n {
2255     ipa~ sans~
2256     upright~ features           = {
2257       IgnoreFontspecFile,
2258       StylisticSet             = { 5 }
2259     }
2260   }
2261 }
2262 \tl_if_in:cnT {
2263   g _ lngx _ ipa _ mono _ upright _ tl
2264 } { NewCM } {
2265   \lngx_set_keys:n {
2266     ipa~ mono~
2267     upright~ features           = {
2268       IgnoreFontspecFile,
2269       StylisticSet             = { 5 }
2270     }
2271   }
2272 }
2273 \lngx_set_main_ipa_font:ee {
2274   \g_lngx_ipa_upright_features_tl
2275 } {
2276   \g_lngx_ipa_upright_tl
2277 }
2278 \lngx_set_sans_ipa_font:ee {
2279   \g_lngx_ipa_sans_upright_features_tl
2280 } {
2281   \g_lngx_ipa_sans_upright_tl
2282 }
2283 \lngx_set_mono_ipa_font:ee {
2284   \g_lngx_ipa_mono_upright_features_tl
2285 } {
2286   \g_lngx_ipa_mono_upright_tl
2287 }
2288 }
2289 </ipa>

```

```

2290 <*logos>
2291 \ProvidesExplPackage{linguistix-logos}
2292     {2025-05-29}
2293     {v0.2}
2294     {%
2295         Logos of the ‘LinguisTiX’ bundle..%
2296     }

```

The fontspec package (if not already loaded).

```

2297
2298 \IfPackageLoadedF { fontspec } {
2299     \RequirePackage { fontspec }
2300 }

```

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

```

2301
2302 \newfontfamily \lngx_logo_font: [
2303     IgnoreFontspecFile,
2304     UprightFont           = { NewCMUncial10-Book.otf },
2305     UprightFeatures       = {
2306         SizeFeatures      = {
2307             {
2308                 Size        = {-8},
2309                 Font        = {NewCMUncial08-Book.otf}
2310             },
2311             {
2312                 Size        = {8-},
2313                 Font        = {NewCMUncial10-Book.otf}
2314             },
2315         }
2316     },
2317     BoldFont              = { NewCMUncial10-Bold.otf },
2318     BoldFeatures          = {
2319         SizeFeatures      = {
2320             {
2321                 Size        = {-8},
2322                 Font        = {NewCMUncial08-Bold.otf}
2323             },
2324             {
2325                 Size        = {8-},
2326                 Font        = {NewCMUncial10-Bold.otf}
2327             },
2328         }
2329     },
2330     Renderer              = { HarfBuzz }
2331 ]{ NewCMUncial10-Book.otf }

```

(End of definition for \lngx_logo_font:. This function is documented on page 14.)

lngx_purple_color

```

2332
2333 \color_set:nn { lngx _ purple _ color } { blue ! 50 ! red }

```

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

\lngxlogo

```
2334
2335 \NewDocumentCommand \lngxlogo { 0{ } } {%
2336   \group_begin:
2337   \lngx_logo_font:
2338   LinguisTi
2339   \color_group_begin:
2340   \color_select:n { lngx_purple_color }
2341   X
2342   \color_group_end:
2343   \IfBlankF { #1 } { - #1 }
2344   \group_end:
2345 }
```

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

\lngxpkg

```
2346
2347 \cs_new:Npn \lngxpkg {
2348   \IfPackageLoadedTF { hyperref } {
2349     \texorpdfstring {
2350       \lngxlogo
2351     } {
2352       LinguisTiX
2353     }
2354   } {
2355     \lngxlogo
2356   }
2357 }
```

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

\lngxbaselogo

\lngxfontslogo

\lngxipalogo

\lngxlogoslogo

\lngxnfsslogo

```
2358
2359 \clist_map_inline:nn {
2360   base,
2361   examples,
2362   fonts,
2363   ipa,
2364   logos,
2365   nfss
2366 } {
2367   \cs_new:cpn { lngx #1 logo } {
2368     \texorpdfstring {
2369       \lngxlogo [ #1 ]
2370     } {
2371       LinguisTiX - #1
2372     }
2373   }
2374 }
2375 </logos>
```

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

```

2376 <nfss>
2377 \ProvidesExplPackage{linguistix-nfss}
2378 {2025-05-29}
2379 {v0.2}
2380 {%
2381   An extension to the core NFSS commands
2382   from the ‘LinguisTiX’ bundle.%
2383 }

```

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.

```

2384
2385 \tl_new:N \l__lngx_normalfont_tmp_tl
2386 \tl_new:N \l__lngx_selectfont_tmp_tl
2387 \tl_new:N \l__lngx_family_tmp_tl
2388 \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.

```

2389
2390 \tl_new:N \l_lngx_current_encoding_tl
2391 \tl_new:N \l_lngx_current_meta_family_tl
2392 \tl_new:N \l_lngx_current_super_family_tl
2393 \tl_new:N \l_lngx_current_series_tl
2394 \tl_new:N \l_lngx_current_shape_tl

```

```

\c_lngx_default_rmdefault_tl
\c_lngx_default_sfdefault_tl
\c_lngx_default_ttdefault_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.

```

2395
2396 \hook_gput_code:nnn { begindocument / end } { . } {
2397   \tl_const:Ne \c_lngx_default_rmdefault_tl { \rmdefault }
2398   \tl_const:Ne \c_lngx_default_sfdefault_tl { \sfdefault }
2399   \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.

```

2400 \tl_set:Nn \l_lngx_current_super_family_tl { default }

```

The current encoding is saved in the relevant t_l.

```

2401 \tl_set:Ne \l_lngx_current_encoding_tl {
2402   \encodingdefault
2403 }

```

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.

```

2404 \IfClassLoadedTF { beamer } {
2405   \tl_set:Ne \l_lngx_current_meta_family_tl { sf }
2406 } {
2407   \tl_set:Ne \l_lngx_current_meta_family_tl { rm }
2408 }

```

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

```

2409 \tl_set:Nc \l_lngx_current_series_tl { md }
2410 \tl_set:Nc \l_lngx_current_shape_tl { up }
2411 }

```

(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.

```

2412
2413 \hook_gput_code:nnn { cmd / normalfont / before } { . } {
2414   \tl_set:Nc \l__lngx_normalfont_tmp_tl { \f@encoding }
2415 }

```

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`.

```

2416
2417 \hook_gput_code:nnn { cmd / normalfont / after } { . } {
2418   \tl_set_eq:NN \l_lngx_current_encoding_tl
2419     \l__lngx_normalfont_tmp_tl
2420   \tl_clear:N \l__lngx_normalfont_tmp_tl
2421 }

```

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

```

2422
2423 \hook_gput_code:nnn { cmd / selectfont / before } { . } {
2424   \tl_set:Nc \l__lngx_selectfont_tmp_tl { \f@encoding }
2425 }
2426
2427 \hook_gput_code:nnn { cmd / selectfont / after } { . } {
2428   \tl_set_eq:NN \l_lngx_current_encoding_tl
2429     \l__lngx_selectfont_tmp_tl
2430   \tl_clear:N \l__lngx_selectfont_tmp_tl
2431 }

```

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.

```

2432
2433 \hook_gput_code:nnn { cmd / rmfamily / before } { . } {
2434   \tl_set:Nc \l_lngx_current_meta_family_tl { rm }
2435   \tl_set:Nc \l__lngx_family_tmp_tl { \f@encoding }
2436 }
2437
2438 \hook_gput_code:nnn { cmd / rmfamily / after } { . } {
2439   \tl_set:Nc \l_lngx_current_meta_family_tl { rm }
2440   \tl_set_eq:NN \l_lngx_current_encoding_tl
2441     \l__lngx_family_tmp_tl
2442   \tl_clear:N \l__lngx_family_tmp_tl
2443 }
2444
2445 \hook_gput_code:nnn { cmd / sffamily / before } { . } {
2446   \tl_set:Nc \l_lngx_current_meta_family_tl { sf }

```

```

2447 \tl_set:Nc \l__lngx_family_tmp_tl { \f@encoding }
2448 }
2449
2450 \hook_gput_code:nnn { cmd / sffamily / after } { . } {
2451   \tl_set:Nn \l_lngx_current_meta_family_tl { sf }
2452   \tl_set_eq:NN \l_lngx_current_encoding_tl
2453     \l__lngx_family_tmp_tl
2454   \tl_clear:N \l__lngx_family_tmp_tl
2455 }
2456
2457 \hook_gput_code:nnn { cmd / ttfamily / before } { . } {
2458   \tl_set:Nn \l_lngx_current_meta_family_tl { tt }
2459   \tl_set:Nc \l__lngx_family_tmp_tl { \f@encoding }
2460 }
2461
2462 \hook_gput_code:nnn { cmd / ttfamily / after } { . } {
2463   \tl_set:Nn \l_lngx_current_meta_family_tl { tt }
2464   \tl_set_eq:NN \l_lngx_current_encoding_tl
2465     \l__lngx_family_tmp_tl
2466   \tl_clear:N \l__lngx_family_tmp_tl
2467 }

```

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 `bf` is more intuitive, plus they also can be used in the argument of `\use:c` directly.

```

2468
2469 \hook_gput_code:nnn { cmd / mdseries / after } { . } {
2470   \tl_set:Nn \l_lngx_current_series_tl { md }
2471 }
2472
2473 \hook_gput_code:nnn { cmd / bfseries / after } { . } {
2474   \tl_set:Nn \l_lngx_current_series_tl { bf }
2475 }

```

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

```

2476
2477 \hook_gput_code:nnn { cmd / upshape / after } { . } {
2478   \tl_set:Nn \l_lngx_current_shape_tl { up }
2479 }
2480
2481 \hook_gput_code:nnn { cmd / itshape / after } { . } {
2482   \tl_set:Nn \l_lngx_current_shape_tl { it }
2483 }
2484
2485 \hook_gput_code:nnn { cmd / scshape / after } { . } {
2486   \tl_set:Nn \l_lngx_current_shape_tl { sc }
2487 }
2488
2489 \hook_gput_code:nnn { cmd / sscshape / after } { . } {
2490   \tl_set:Nn \l_lngx_current_shape_tl { ssc }
2491 }
2492
2493 \hook_gput_code:nnn { cmd / slshape / after } { . } {
2494   \tl_set:Nn \l_lngx_current_shape_tl { sl }

```

```

2495 }
2496
2497 \hook_gput_code:nnn { cmd / swshape / after } { . } {
2498   \tl_set:Nn \l_lngx_current_shape_tl { sw }
2499 }
2500
2501 \hook_gput_code:nnn { cmd / ulcshape / after } { . } {
2502   \tl_set:Nn \l_lngx_current_shape_tl { ulc }
2503 }
2504
2505 \hook_gput_code:nnn { cmd / ulcshape / after } { . } {
2506   \tl_set:Nn \l_lngx_current_shape_tl { #1 }
2507 }

```

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

```

2508
2509 \prg_new_conditional:Nnn \lngx_if_encoding:n {
2510   p,
2511   T,
2512   F,
2513   TF
2514 } {
2515   \tl_if_eq:NnTF \l_lngx_current_encoding_tl { #1 } {
2516     \prg_return_true:
2517   } {
2518     \prg_return_false:
2519   }
2520 }
2521

```

(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.

`\IfEncodingT`

`\IfEncodingF`

```

2522
2523 \cs_new_eq:NN \IfEncodingTF \lngx_if_encoding:nTF
2524 \cs_new_eq:NN \IfEncodingT \lngx_if_encoding:nT
2525 \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.

`\lngx_if_meta_family:nTF`

```

2526
2527 \prg_new_conditional:Nnn \lngx_if_meta_family:n {
2528   p,
2529   T,
2530   F,
2531   TF
2532 } {
2533   \tl_if_eq:NnTF \l_lngx_current_meta_family_tl { #1 } {
2534     \prg_return_true:
2535   } {
2536     \prg_return_false:
2537   }
2538 }

```

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

`\IfMetaFamilyTF` User-facing conditionals for meta family.

`\IfMetaFamilyT`

`\IfMetaFamilyF`

2539

2540 `\cs_new_eq:NN \IfMetaFamilyTF \lngx_if_meta_family:nTF`

2541 `\cs_new_eq:NN \IfMetaFamilyT \lngx_if_meta_family:nT`

2542 `\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`

2543

2544 `\prg_new_conditional:Nnn \lngx_if_super_family:n {`

2545 `P,`

2546 `T,`

2547 `F,`

2548 `TF`

2549 `} {`

2550 `\tl_if_eq:NnTF \l_lngx_current_super_family_tl { #1 } {`

2551 `\prg_return_true:`

2552 `} {`

2553 `\prg_return_false:`

2554 `}`

2555 `}`

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

`\IfSuperFamilyTF`

User-facing conditionals for super family.

`\IfSuperFamilyT`

`\IfSuperFamilyF`

2556

2557 `\cs_new_eq:NN \IfSuperFamilyTF \lngx_if_super_family:nTF`

2558 `\cs_new_eq:NN \IfSuperFamilyT \lngx_if_super_family:nT`

2559 `\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`

2560

2561 `\prg_new_conditional:Nnn \lngx_if_series:n {`

2562 `P,`

2563 `T,`

2564 `F,`

2565 `TF`

2566 `} {`

2567 `\tl_if_eq:NnTF \l_lngx_current_series_tl { #1 } {`

2568 `\prg_return_true:`

2569 `} {`

2570 `\prg_return_false:`

2571 `}`

2572 `}`

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

`\IfSeriesTF` Its user-side macros.

```
\IfSeriesT      2573
\IfSeriesF      2574 \cs_new_eq:NN \IfSeriesTF \lngx_if_series:nTF
                  2575 \cs_new_eq:NN \IfSeriesT  \lngx_if_series:nT
                  2576 \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 2577
                    2578 \prg_new_conditional:Nnn \lngx_if_shape:n {
                    2579   p,
                    2580   T,
                    2581   F,
                    2582   TF
                    2583 } {
                    2584   \tl_if_eq:NnTF \l_lngx_current_shape_tl { #1 } {
                    2585     \prg_return_true:
                    2586   } {
                    2587     \prg_return_false:
                    2588   }
                    2589 }
```

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

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

```
\IfShapeT      2590
\IfShapeF      2591 \cs_new_eq:NN \IfShapeTF \lngx_if_shape:nTF
                  2592 \cs_new_eq:NN \IfShapeT  \lngx_if_shape:nT
                  2593 \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.

```
2594
2595 \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: 2596
\lngx_if_meta_family_sf:TF 2597 \clist_map_inline:nn {
\lngx_if_meta_family_tt_p: 2598   rm,
\lngx_if_meta_family_tt:TF 2599   sf,
                             2600   tt
                             2601 } {
                             2602   \prg_new_conditional:cnn {
                             2603     lngx _ if _ meta _ family _ #1 :
                             2604   } {
                             2605     p, T, F, TF
                             2606   } {
                             2607     \tl_if_eq:NnTF \l_lngx_current_meta_family_tl { #1 } {
                             2608       \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.)

```
\ltx_if_series_md:TF
\ltx_if_series_bf_p:
\ltx_if_series_bf:TF
```

(End of definition for `\lngx_if_series_md:TF` and `\lngx_if_series_bf:TF`. These functions are documented on page 15.)

```

\ltx_if_shape_up:TF
\ltx_if_shape_it_p:
\ltx_if_shape_it:TF
\ltx_if_shape_sc_p:
\ltx_if_shape_sc:TF
\ltx_if_shape_ssc_p:
\ltx_if_shape_ssc:TF
\ltx_if_shape_sl_p:
\ltx_if_shape_sl:TF
\ltx_if_shape_sw_p:
\ltx_if_shape_sw:TF
\ltx_if_shape_ulc_p:
\ltx_if_shape_ulc:TF

```

(End of definition for `\ltx_if_shape_up:TF` and others. These functions are documented on page 16.)

74

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.

```

2649
2650 \clist_map_inline:nn {
2651   rm,
2652   sf,
2653   tt
2654 } {
2655   \keys_define:nn { lngx _ nfss } {
2656     #1
2657     .code:n          = {
2658       \tl_gclear_new:c {
2659         g _ lngx _ \l__lngx_nfss_tmp_tl _ #1 default _ tl
2660       }
2661       \tl_gset:cn {
2662         g _ lngx _ \l__lngx_nfss_tmp_tl _ #1 default _ tl
2663       } { ##1 }
2664     }
2665   }
2666 }

```

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

```

2667
2668 \cs_new_protected:Npn \lngx_super_font_family:nn #1#2 {
2669   \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.

```

2670   \keys_set:nn { lngx _ nfss } { #2 }
2671   \tl_clear:N \l__lngx_nfss_tmp_tl
2672 }
2673
2674 \cs_set_eq:NN \superfontfamily
2675           \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.
`\softsuperfontfamily`

```

2676
2677 \cs_new_protected:Npn \lngx_soft_super_font_family:nn #1#2 {
2678   \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.

```

2679   \clist_map_inline:nn {
2680     rm,
2681     sf,
2682     tt
2683   } {
2684     \tl_if_empty:cF { g _ lngx _ #1 _ ##1 default _ tl } {
2685       \cs_set:cpe { ##1 default } {

```

```

2686         \tl_use:c { g _ lngx _ #1 _ ##1 default _ tl }
2687     }
2688 }
2689 }

```

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

```

2690 \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.

```

2691 \clist_map_inline:nn { #2 } {
2692     \str_case:nn { ##1 } {
2693         { encoding } {
2694             \exp_args:NV \fontencoding
2695                 \l_lngx_current_encoding_tl
2696         }
2697         { family } {
2698             \use:c {
2699                 \l_lngx_current_meta_family_tl family
2700             }
2701             \exp_args:NV \fontencoding
2702                 \l_lngx_current_encoding_tl
2703             \selectfont
2704         }
2705         { series } {
2706             \use:c {
2707                 \l_lngx_current_series_tl series
2708             }
2709         }
2710         { shape } {
2711             \use:c {
2712                 \l_lngx_current_shape_tl shape
2713             }
2714         }
2715     }
2716 }
2717 }
2718
2719 \cs_set_eq:NN \softsuperfontfamily
2720             \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.

```

2721
2722 \cs_new_protected:Npn \lngx softer super font family:n #1 {
2723     \lngx_soft_super_font_family:nn { #1 } {
2724         family,
2725         series,
2726         shape

```

```

2727 }
2728 }
2729
2730 \cs_set_eq:NN \softersuperfontfamily
2731 \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.

```

2732
2733 \cs_new_protected:Npn \lngx_softest_super_font_family:n #1 {
2734   \lngx_soft_super_font_family:nn { #1 } {
2735     encoding,
2736     family,
2737     series,
2738     shape
2739   }
2740 }
2741
2742 \cs_set_eq:NN \softestsuperfontfamily
2743 \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.

```

2744
2745 \cs_new_protected:Npn \lngx_soft_normal_font:n #1 {
2746   \tl_set:Nx \l_lngx_current_super_family_tl { default }
2747   \clist_map_inline:nn {
2748     rm,
2749     sf,
2750     tt
2751   } {
2752     \cs_set:cpe { ##1 default } {
2753       \tl_use:c { c _ lngx _ default _ ##1 default _ tl }
2754     }
2755   }
2756   \normalfont
2757   \clist_map_inline:nn { #1 } {
2758     \str_case:nn { ##1 } {
2759       { encoding } {
2760         \exp_args:NV \fontencoding
2761           \l_lngx_current_encoding_tl
2762       }
2763       { family } {
2764         \use:c {
2765           \l_lngx_current_meta_family_tl family
2766         }
2767         \exp_args:NV \fontencoding
2768           \l_lngx_current_encoding_tl
2769       }
2770     }

```

```

2771     { series } {
2772       \use:c {
2773         \l_lngx_current_series_tl series
2774       }
2775     }
2776     { shape } {
2777       \use:c {
2778         \l_lngx_current_shape_tl shape
2779       }
2780     }
2781   }
2782 }
2783 }
2784
2785 \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_softest_normal_font:` This is a parallel to the ‘softer’ super family command for the default super family.
`\softernormalfont`

```

2786
2787 \cs_new_protected:Npn \lngx_softest_normal_font: {
2788   \lngx_soft_normal_font:n {
2789     family,
2790     series,
2791     shape
2792   }
2793 }
2794
2795 \cs_set_eq:NN \softernormalfont \lngx_softest_normal_font:

```

(End of definition for `\lngx_softest_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`

```

2796
2797 \cs_new_protected:Npn \lngx_softest_normal_font: {
2798   \lngx_soft_normal_font:n {
2799     encoding,
2800     family,
2801     series,
2802     shape
2803   }
2804 }
2805
2806 \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.

```

2807 \cs_new:Npn \CurrentEncoding {
2808   \tl_use:N \l_lngx_current_encoding_tl
2809 }

```

```

2810 \cs_new:Npn \CurrentMetaFamily {
2811   \tl_use:N \l_lngx_current_meta_family_tl
2812 }
2813 \cs_new:Npn \CurrentSuperFamily {
2814   \tl_use:N \l_lngx_current_super_family_tl
2815 }
2816 \cs_new:Npn \CurrentSeries {
2817   \tl_use:N \l_lngx_current_series_tl
2818 }
2819 \cs_new:Npn \CurrentShape {
2820   \tl_use:N \l_lngx_current_shape_tl
2821 }
2822 \</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.

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.