

The LINGUISṬIX bundle

निरंजन

May 20, 2025 (v0.1b)

🏠 <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	6
2	Planned	4		Interface... 12; Implementation... 41	
3	Funding	4	8	LINGUISṬIX-Logos	8
4	Acknowledgements	4		Interface... 14; Implementation... 66	
5	LINGUISṬIX-BASE	4	9	LINGUISṬIX-NFSS	9
	Interface... 12; Implementation... 18			Interface... 14; Implementation... 68	
6	LINGUISṬIX-FONTS	5		GNU Free Documentation License	80
	Interface... 12; 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 [a], but unfortunately, in Italic shape, the latter is a variant of the former. E.g., `[a\textit{a}]` produces '[aa]'. Whenever an author uses Italic shape for their transcription and use `a`, a wrong IPA symbol is printed with most fonts. This problem was kindly acknowledged by Antonis Tsolomitis, 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 [gg].

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

2 Planned

I plan to develop this bundle further in order to support the typesetting of good quality examples with interlinear glossing. My model is to imitate the output of the `expex` package, but with a modern L^AT_EX-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_{TI}X 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.

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 L^AT_EX₃’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 L^AT_EX. Lastly, to all the free software people who have created this friendly and supportive world for people by investing their precious time and resources!

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_{TI}X 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_{TI}X-BASE

L^AT_EX₃-interface | Implementation

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

`\linguistix {⟨key-value-list⟩}`

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

6 LINGUIS \mathcal{T} X-FONTS

L^AT_EX₃-interface | **Implementation**

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

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

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

<code>old style numbers</code>	<code>= {⟨truth value⟩}</code>	<code>true</code> <code>false</code>
<code>old style one</code>	<code>= {⟨truth value⟩}</code>	<code>true</code> <code>false</code>

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

Suppose one wants the alternative shape of number 'r' 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
0123456789	Old style with the old 1
0123456789	Lining

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

```
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

LINGUIS $\overline{\text{T}}$ X-FONTS sets maths fonts also. In order to control the settings related to maths, the following keys can be used.

```
math = {\math font}
math features = {\math font features}
math bold = {\bold math font}
math bold features = {\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.

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

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

7 LINGUIS $\overline{\text{T}}$ X-ipa

L^AT_EX₃-interface | Implementation

This package sets the fonts exclusively for the IPA. The commands provided for switching to the IPA control all serif, sans serif and typewriter families. This package can be loaded standalone for loading IPA fonts as well as some switch commands useful in running text. New Computer Modern provides a special stylistic set dedicated for linguistics. It is enabled for IPA fonts automatically with this package. Only the legally marked up IPA is affected by the customisation provided by this package. For switching to the IPA, LINGUIS $\overline{\text{T}}$ 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{a1 phi: e1}` \longrightarrow $[a_1 p^{hi}: e_1]$ whereas the starred version prints it in slashes for phonemic transcription, e.g.: `\ipatext*{a1 phi: e1}` \longrightarrow $/a_1 p^{hi}: e_1/$.

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

`\lngxipa` This also is a command that switches to the IPA-only features (default as well as user added). This command, of course, leaks and that's why *should* be delimited. E.g., the following code lines produce [a_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 LINGUISCIX-FONTS and LINGUISCIX-IPA.

Family	LINGUISCIX-FONTS	LINGUISCIX-IPA
Serif	text upright	ipa upright
	text upright features	ipa upright features
	text bold upright	ipa bold upright
	text bold upright features	ipa bold upright features
	text italic	ipa italic
	text italic features	ipa italic features
	text bold italic	ipa bold italic
	text bold italic features	ipa bold italic features
	text slanted	ipa slanted
	text slanted features	ipa slanted features
	text bold slanted	ipa bold slanted
	text bold slanted features	ipa bold slanted features
	text swash	ipa swash
	text swash features	ipa swash features
	text bold swash	ipa bold swash
	text bold swash features	ipa bold swash features
	text small caps	ipa small caps
	text small caps features	ipa small caps features
Sans serif	text sans upright	ipa sans upright
	text sans upright features	ipa sans upright features
	text sans bold upright	ipa sans bold upright
	text sans bold upright features	ipa sans bold upright features
	text sans italic	ipa sans italic
	text sans italic features	ipa sans italic features

Continued on the next page...

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

8 LINGUIS \mathcal{T} \mathbf{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} \mathbf{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} \mathbf{I} \mathbf{X} bundle is printed with this command, e.g., `\lngxlogo{fonts} → LINGUIS \mathcal{T} \mathbf{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 \mathcal{X}
<code>\lngxbaselogo</code>	★	LINGUIS \mathcal{T} I \mathcal{X} -BASE
<code>\lngxfontlogo</code>	★	LINGUIS \mathcal{T} I \mathcal{X} -FONTS
<code>\lngxipalogo</code>	★	LINGUIS \mathcal{T} I \mathcal{X} -IPA
<code>\lngxlogoslogo</code>	★	LINGUIS \mathcal{T} I \mathcal{X} -LOGOS
<code>\lngxnfsslogo</code>	★	LINGUIS \mathcal{T} I \mathcal{X} -NFSS

9 LINGUIS \mathcal{T} I \mathcal{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 LuaL^AT_EX.

```

\documentclass{article}

\begin{document}
\makeatletter
\fontencoding{OT1}\sffamily\itshape\bfseries
\selectfont
\fontencoding\ | \f@family\ | \f@series\ | \f@shape\quad
\normalfont
\fontencoding\ | \f@family\ | \f@series\ | \f@shape
\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-NFSS` is useful in such cases. It introduces the concept of ‘super’ font family. It shouldn’t be confused with \LaTeX 2_ϵ ’s ‘meta’ font family. It refers to `rm`, `sf` or `tt` in the kernel. Note that, as of now, \LaTeX 2_ϵ does *not* provide any public interface to save ‘meta’ family, as well as, the current encoding, series and shape. This package provides control over these facets. Let’s have a look at the macros it provides.

<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>	<code>* If the current encoding matches with the given $\langle encoding \rangle$, it selects the true branch; false otherwise. The <code>\CurrentEncoding</code> macro expands to the current encoding.</code>

<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>	<code>* If the current meta family matches with the given $\langle meta family \rangle$, it selects the true branch; false otherwise. The <code>\CurrentMetaFamily</code> macro expands to the current meta family.</code>

<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>	<code>* If the current super family matches with the given $\langle super family \rangle$, it selects the true branch; false otherwise. The <code>\CurrentSuperFamily</code> macro expands to the current super family.</code>

<code>\IfSeriesTF</code>	<code>*</code>	<code>{\langle series \rangle}{\langle true code \rangle}{\langle false code \rangle}</code>
<code>\IfSeriesT</code>	<code>*</code>	<code>{\langle series \rangle}{\langle true code \rangle}</code>
<code>\IfSeriesF</code>	<code>*</code>	<code>{\langle series \rangle}{\langle false code \rangle}</code>

`\CurrentSeries` `*` If the current series matches with the given $\langle series \rangle$, it selects the true branch and false otherwise. The `\CurrentSeries` macro expands to the current series.

<code>\IfShapeTF</code>	<code>*</code>	<code>{\langle shape \rangle}{\langle true code \rangle}{\langle false code \rangle}</code>
<code>\IfShapeT</code>	<code>*</code>	<code>{\langle shape \rangle}{\langle true code \rangle}</code>
<code>\IfShapeF</code>	<code>*</code>	<code>{\langle shape \rangle}{\langle false code \rangle}</code>

`\CurrentShape` `*` If the current series matches with the given $\langle shape \rangle$, it selects the true branch and false otherwise. The `\CurrentShape` macro expands to the current shape.

`\superfontfamily` `{\langle family id \rangle}{\langle rm=\langle rm nfss \rangle, sf=\langle sf nfss \rangle, tt=\langle tt nfss \rangle \rangle}`

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

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

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

<code>\softnormalfont</code>	<code>{\langle encoding, family, series, shape \rangle}</code>
<code>\softernormalfont</code>	Similar to <code>\softsuperfontfamily</code> and friends, these commands switch back to the default super font family, but reactivate the previously active font attributes. The argument to <code>\softnormalfont</code> takes the list of the required font attributes. It can have ≤ 4 values. Now try the following example:
<code>\softestnormalfont</code>	

```

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

\begin{document}
test \lmgxipa test \softernormalfont test\par
\makeatletter
\sffamily\itshape\bfseries

```

```

\fontfamily\ | \fontseries\ | \fontshape\quad
\softnormalfont{series}
\fontfamily\ | \fontseries\ | \fontshape
\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.

LINGUIS_TI_X-BASE

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

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

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

LINGUIS_TI_X-FONTS

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

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

<code>\lngx_set_main_font:nn</code>	<code>\lngx_set_main_font:nn</code> $\{\langle features \rangle\}$ $\{\langle font \rangle\}$
<code>\lngx_set_main_font:ee</code>	<code>\lngx_set_sans_font:nn</code> $\{\langle features \rangle\}$ $\{\langle font \rangle\}$
<code>\lngx_set_sans_font:nn</code>	<code>\lngx_set_mono_font:nn</code> $\{\langle features \rangle\}$ $\{\langle font \rangle\}$
<code>\lngx_set_sans_font:ee</code>	<code>\lngx_set_math_font:nn</code> $\{\langle features \rangle\}$ $\{\langle font \rangle\}$
<code>\lngx_set_mono_font:nn</code>	
<code>\lngx_set_mono_font:ee</code>	
<code>\lngx_set_math_font:nn</code>	
<code>\lngx_set_math_font:ee</code>	

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 $\langle features \rangle$ are passed to the optional argument and the $\langle font \rangle$ is passed to the mandatory argument of the respective command from the aforementioned list.

LINGUIS_TI_X-IPA

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

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

<code>\lngx_set_main_ipa_font:nn</code>	<code>\lngx_set_main_ipa_font:nn</code> $\{\langle features \rangle\}$ $\{\langle font \rangle\}$
<code>\lngx_set_main_ipa_font:ee</code>	
<code>\lngx_main_ipa:</code>	
<code>lngx_ipa_rm_nfss</code>	

These functions set the IPA fonts for the serif variants. The $\langle font \rangle$ is set with $\langle features \rangle$ for the serif IPA. The command to switch to this family is `\lngx_main_ipa:.` It can be accessed with the NFSS family `lngx_ipa_rm_nfss`.

<hr/> <code>\lngx_set_sans_ipa_font:nn</code>	<code>\lngx_set_sans_ipa_font:nn {\features} {\font}</code>
<code>\lngx_set_sans_ipa_font:ee</code>	
<code>\lngx_sans_ipa:</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
<code>lngx_ipa_sf_nfss</code>	accessed with the NFSS family <code>lngx_ipa_sf_nfss</code> .

<hr/> <code>\lngx_set_mono_ipa_font:nn</code>	<code>\lngx_set_mono_ipa_font:nn {\features} {\font}</code>
<code>\lngx_set_mono_ipa_font:ee</code>	
<code>\lngx_mono_ipa:</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
<code>lngx_ipa_tt_nfss</code>	accessed with the NFSS family <code>lngx_ipa_nfss_nfss</code> .

<hr/> <code>\lngx_ipa:</code>	The <code>\lngx_ipa:</code> command loads the super family <code>lngx_ipa</code> (see the documentation of
<code>lngx_ipa</code>	<code>LINGUISCTIX-NFSS</code> . 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 `LINGUISCTIX-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>

Continued on the next page...

Serif	Sans serif	Monospaced
<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 `LINGUIS \mathcal{T} \mathbf{I} \mathbf{X} -FONTS` and `LINGUIS \mathcal{T} \mathbf{I} \mathbf{X} -IPA`

LINGUIS \mathcal{T} \mathbf{I} \mathbf{X} -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 \mathcal{T} \mathbf{I} \mathbf{X} -NFSS

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

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

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

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

```

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

```

```

\lngx_if_meta_family_rm_p: *
\lngx_if_meta_family_rm:TF * {\true code}{\false code}
\lngx_if_meta_family_sf_p: *
\lngx_if_meta_family_sf:TF * {\true code}{\false code}
\lngx_if_meta_family_tt_p: *
\lngx_if_meta_family_tt:TF * {\true code}{\false code}

```

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 * {\true code}{\false code}
\lngx_if_series_bf_p: *
\lngx_if_series_bf:TF * {\true code}{\false code}

```

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

```

This function takes an **<id>** 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.

LINGUISTIX

Provide the package with its basic information.

```
1 <*package>
2 \ProvidesExplPackage{linguistix}
3     {2025-05-20}
4     {v0.1b}
5     {%
6         The ‘Linguistix’ bundle: Enhanced
7         support for linguistics.%
8     }
```

When one loads LINGUISTIX, all the packages of the bundle are loaded automatically. That’s the only content of the umbrella package LINGUISTIX. 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     {2025-05-20}
29     {v0.1b}
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     {2025-05-20}
44     {v0.1b}
45     {%
46         The font-assistant package of the
47         'Linguistix' bundle.%
48     }

```

I load LINGUIS**TiX**-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 }

```

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

```

57
58 \keys_define:nn { lngx _ keys } {
59     old~ style~ numbers
60     .bool_gset:N          = {
61         \g_lngx_old_style_bool
62     },
63     old~ style~ one
64     .bool_gset:N          = {
65         \g_lngx_old_style_one_bool
66     },
67     bourbaki's~ empty~ set
68     .bool_gset:N          = {
69         \g_lngx_bourbaki_bool
70     }
71 }

```

(End of definition for *old style numbers* and others. These functions are documented on page 5.)

```

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>.)

```

72
73 \clist_map_inline:nn {
74   upright,
75   bold~ upright,
76   italic,
77   bold~ italic,
78   slanted,
79   bold~ slanted,
80   swash,
81   bold~ swash,
82   small~ caps
83 } {

```

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.

```

84 \tl_set:Nn \l_tmpa_tl { #1 }
85 \tl_replace_all:Nnn \l_tmpa_tl { ~ } { _ }
86 \tl_gclear_new:c {
87   g _ lngx _ text _ \l_tmpa_tl _ features _ tl
88 }

```

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`.

```

89 \keys_define:nn { lngx _ keys } {
90   text~ #1
91   .tl_gset_e:c = {
92     g _ lngx _ text _ \l_tmpa_tl _ tl
93   },

```

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.

```

94   text~ #1~ features
95   .code:n = {
96     \tl_set:Nn \l_tmpb_tl { #1 }
97     \tl_replace_all:Nnn \l_tmpb_tl { ~ } { _ }
98     \tl_put_right:ce {
99       g _ lngx _ text _ \l_tmpb_tl _ features _ tl
100     } { ##1 , }

```

Lastly, we clear the temporary `tl`s.

```

101   \tl_clear:N \l_tmpb_tl
102 }
103 }
104 \tl_clear:N \l_tmpa_tl
105 }

```

(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.

```

106
107 \clist_map_inline:nn {
108   upright,
109   bold~ upright,
110   italic,
111   bold~ italic,
112   slanted,
113   bold~ slanted,
114   swash,
115   bold~ swash,
116   small~ caps
117 } {
118   \tl_set:Nn \l_tmpa_tl { #1 }
119   \tl_replace_all:Nnn \l_tmpa_tl { ~ } { _ }
120   \tl_gclear_new:c {
121     g _ lngx _ text _ sans _ \l_tmpa_tl _ features _ tl
122   }
123   \keys_define:nn { lngx _ keys } {
124     text~ sans~ #1
125     .tl_gset_e:c      = {
126       g _ lngx _ text _ sans _ \l_tmpa_tl _ tl
127     },
128     text~ sans~ #1~ features
129     .code:n           = {
130       \tl_set:Nn \l_tmpb_tl { #1 }
131       \tl_replace_all:Nnn \l_tmpb_tl { ~ } { _ }
132       \tl_put_right:ce {
133         g _ lngx _ text _ sans _ \l_tmpb_tl _ features _ tl
134       } { ##1 , }
135       \tl_clear:N \l_tmpb_tl
136     }
137   }
138   \tl_clear:N \l_tmpa_tl
139 }

```

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

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

```

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

```

```

I40
I41 \clist_map_inline:nn {
I42   upright,
I43   bold~ upright,
I44   italic,
I45   bold~ italic,
I46   slanted,
I47   bold~ slanted,
I48   swash,
I49   bold~ swash,
I50   small~ caps
I51 } {
I52   \tl_set:Nn \l_tmpa_tl { #1 }
I53   \tl_replace_all:Nnn \l_tmpa_tl { ~ } { _ }
I54   \tl_gclear_new:c {
I55     g _ lngx _ text _ mono _ \l_tmpa_tl _ features _ tl
I56   }
I57   \keys_define:nn { lngx _ keys } {
I58     text~ mono~ #1
I59     .tl_gset_e:c          = {
I60       g _ lngx _ text _ mono _ \l_tmpa_tl _ tl
I61     },
I62     text~ mono~ #1~ features
I63     .code:n              = {
I64       \tl_set:Nn \l_tmpb_tl { #1 }
I65       \tl_replace_all:Nnn \l_tmpb_tl { ~ } { _ }
I66       \tl_put_right:ce {
I67         g _ lngx _ text _ mono _ \l_tmpb_tl _ features _ tl
I68       } { ##1 , }
I69       \tl_clear:N \l_tmpb_tl
I70     }
I71   }
I72   \tl_clear:N \l_tmpa_tl
I73 }

```

(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.

```

174
175 \keys_define:nn { lngx _ keys } {
176   math
177   .tl_gset_e:c          = {
178     g _ lngx _ math _ tl
179   },
180   math~ features
181   .tl_gset_e:c          = {
182     g _ lngx _ math _ features _ tl
183   },
184   math~ bold
185   .tl_gset_e:c          = {
186     g _ lngx _ math _ bold _ tl
187   },
188   math~ bold~ features
189   .code:n               = {
190     \tl_put_right:ce {
191       g _ lngx _ math _ bold _ features _ tl
192     } { #1 }
193   }
194 }
```

(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.

```

195
196 \keys_define:nn { lngx _ keys } {
197   newcm
198   .meta:n               = {
199     text~
200     upright              = {
201       NewCM10-Book.otf
202     },
203     text~
204     bold~ upright        = {
205       NewCM10-Bold.otf
206     },
207     text~
208     italic               = {
209       NewCM10-BookItalic.otf
210     },
211     text~
212     bold~ italic         = {
213       NewCM10-BoldItalic.otf
214     },
215     math                 = {
216       NewCMMath-Book.otf
217     },
218     math~ bold           = {
219       NewCMMath-Bold.otf
220     },
221     text~
```

```

222 sans~ upright          = {
223     NewCMSans10-Book.otf
224 },
225 text~
226 sans~ bold~ upright    = {
227     NewCMSans10-Bold.otf
228 },
229 text~
230 sans~ italic           = {
231     NewCMSans10-BookOblique.otf
232 },
233 text~
234 sans~ bold~ italic     = {
235     NewCMSans10-BoldOblique.otf
236 },
237 text~
238 mono~ upright          = {
239     NewCMMono10-Book.otf
240 },
241 text~
242 mono~ bold~ upright    = {
243     NewCMMono10-Bold.otf
244 },
245 text~
246 mono~ italic           = {
247     NewCMMono10-BookItalic.otf
248 },
249 text~
250 mono~ bold~ italic     = {
251     NewCMMono10-BoldOblique.otf
252 }
253 }
254 }

```

(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.

```

255
256 \keys_define:nn { lngx _ keys } {
257   newcm~ sans
258   .meta:n          = {
259     text~
260     upright         = {
261         NewCMSans10-Book.otf
262     },
263     text~
264     bold upright    = {
265         NewCMSans10-Bold.otf
266     },
267     text~
268     italic          = {
269         NewCMSans10-BookOblique.otf
270     },
271     text~

```



```

272     bold~ italic          = {
273         NewCMSans10-BoldOblique.otf
274     }
275 }
276 }

```

(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.

```

277
278 \keys_define:nn { lngx _ keys } {
279     newcm~ mono
280     .meta:n          = {
281         text~
282         upright       = {
283             NewCMMono10-Book.otf
284         },
285         text~
286         bold upright  = {
287             NewCMMono10-Bold.otf
288         },
289         text~
290         italic        = {
291             NewCMMono10-BookItalic.otf
292         },
293         text~
294         bold~ italic  = {
295             NewCMMono10-BoldOblique.otf
296         }
297     }
298 }

```

(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.

```

299
300 \keys_define:nn { lngx _ keys } {
301     newcm~ regular
302     .meta:n          = {
303         text~
304         upright       = {
305             NewCM10-Regular.otf
306         },
307         text~
308         bold~ upright  = {
309             NewCM10-Bold.otf
310         },
311         text~
312         italic        = {
313             NewCM10-Italic.otf
314         },
315         text~
316         bold~ italic  = {

```

```

317     NewCM10-BoldItalic.otf
318 },
319 math                = {
320     NewCMMath-Regular.otf
321 },
322 math~ bold          = {
323     NewCMMath-Bold.otf
324 },
325 text~
326 sans~ upright       = {
327     NewCMSans10-Regular.otf
328 },
329 text~
330 sans~ bold~ upright = {
331     NewCMSans10-Bold.otf
332 },
333 text~
334 sans~ italic        = {
335     NewCMSans10-Oblique.otf
336 },
337 text~
338 sans~ bold~ italic  = {
339     NewCMSans10-BoldOblique.otf
340 },
341 text~
342 mono~ upright       = {
343     NewCMMono10-Regular.otf
344 },
345 text~
346 mono~ bold~ upright = {
347     NewCMMono10-Bold.otf
348 },
349 text~
350 mono~ italic        = {
351     NewCMMono10-Italic.otf
352 },
353 text~
354 mono~ bold~ italic  = {
355     NewCMMono10-Bold.otf
356 }
357 }
358 }

```

(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.

```

359 \keys_define:nn { lngx _ keys } {
360   newcm~ regular~ sans
361   .meta:n                = {
362     text~
363     upright               = {
364       NewCMSans10-Regular.otf
365

```

```

366 },
367 text~
368 bold~ upright          = {
369     NewCMSans10-Bold.otf
370 },
371 text~
372 italic                  = {
373     NewCMSans10-Oblique.otf
374 },
375 text~
376 bold~ italic            = {
377     NewCMSans10-BoldOblique.otf
378 }
379 }
380 }

```

(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.

```

381
382 \keys_define:nn { lngx _ keys } {
383     newcm~ regular~ mono
384     .meta:n          = {
385         text~
386         upright       = {
387             NewCMMono10-Regular.otf
388         },
389         text~
390         bold~ upright = {
391             NewCMMono10-Bold.otf
392         },
393         text~
394         italic        = {
395             NewCMMono10-Italic.otf
396         },
397         text~
398         bold~ italic  = {
399             NewCMMono10-Bold.otf
400         }
401     }
402 }

```

(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.

```

403
404 \lngx_set_keys:n {
405     newcm,

```

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

```

406     bourbaki's~ empty~ set,

```

Lastly we load the old `style numbers` boolean.

```
407   old~ style~ numbers
408 }
```

We need HarfBuzz renderer whenever Lua_LAT_EX is used. For that we add the required feature to the feature-lists of all the fonts.

```
409
410 \sys_if_engine luatex:T {
411   \lngx_set_keys:n {
412     text~
413     upright~ features      = {
414       Renderer             = { HarfBuzz }
415     },
416     text~ sans~
417     upright~ features      = {
418       Renderer             = { HarfBuzz }
419     },
420     text~ mono~
421     upright~ features      = {
422       Renderer             = { HarfBuzz }
423     }
424   }
425 }
```

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

```
\lngx_set_sans_font:nn
\lngx_set_mono_font:nn
\lngx_set_math_font:nn
426
427 \cs_new_protected:Npn \lngx_set_main_font:nn #1#2 {
428   \setmainfont [ #1 ] { #2 }
429 }
430
431 \cs_new_protected:Npn \lngx_set_sans_font:nn #1#2 {
432   \setsansfont [ #1 ] { #2 }
433 }
434
435 \cs_new_protected:Npn \lngx_set_mono_font:nn #1#2 {
436   \setmonofont [ #1 ] { #2 }
437 }
438
439 \cs_new_protected:Npn \lngx_set_math_font:nn #1#2 {
440   \setmathfont [ #1 ] { #2 }
441 }
442
443 \cs_generate_variant:Nn \lngx_set_main_font:nn { ee }
444 \cs_generate_variant:Nn \lngx_set_sans_font:nn { ee }
445 \cs_generate_variant:Nn \lngx_set_mono_font:nn { ee }
446 \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 12.)

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.

```

447
448 \hook_gput_code:nnn { begindocument / before } { . } {
449   \tl_if_in:cnT {
450     g _ lngx _ text _ upright _ tl
451   } { 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.

```

452   \tl_if_in:cnTF {
453     g _ lngx _ text _ upright _ tl
454   } { Book } {
455     \lngx_set_keys:n {
456       text~
457       upright~ features      = {
458         SizeFeatures         = {
459           {
460             Size              = {-8},
461             Font              = {
462               NewCM08-Book.otf
463             }
464           },
465           {
466             Size              = {8-},
467             Font              = {
468               NewCM10-Book.otf
469             }
470           }
471         }
472       }
473     }

```

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

```

474   } {
475     \lngx_set_keys:n {
476       text~
477       upright~ features      = {
478         SizeFeatures         = {
479           {
480             Size              = {-8},
481             Font              = {
482               NewCM08-Regular.otf
483             }
484           },
485           {
486             Size              = {8-},
487             Font              = {
488               NewCM10-Regular.otf
489             }
490           }
491         }
492       }

```

```

493     }
494   }
495 }

```

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.

```

496 \tl_if_in:cnT {
497   g _ lngx _ text _ upright _ tl
498 } { NewCMSans } {
499   \tl_if_in:cnTF {
500     g _ lngx _ text _ upright _ tl
501   } { Book } {
502     \lngx_set_keys:n {
503       text~
504       upright~ features      = {
505         SizeFeatures         = {
506           {
507             Size              = {-8},
508             Font              = {
509               NewCMSans08-Book.otf
510             }
511           },
512           {
513             Size              = {8-},
514             Font              = {
515               NewCMSans10-Book.otf
516             }
517           }
518         }
519       }
520     }
521   } {
522     \lngx_set_keys:n {
523       text~
524       upright~ features      = {
525         SizeFeatures         = {
526           {
527             Size              = {-8},
528             Font              = {
529               NewCMSans08-Regular.otf
530             }
531           },
532           {
533             Size              = {8-},
534             Font              = {
535               NewCMSans10-Regular.otf
536             }
537           }
538         }
539       }

```

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.

```

540     }
541   }
542 }

```

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

```

543 \tl_if_in:cnT {
544   g _ lngx _ text _ italic _ tl
545 } { NewCM } {
546   \tl_if_in:cnTF {
547     g _ lngx _ text _ italic _ tl
548   } { Book } {
549     \lngx_set_keys:n {
550       text~
551       italic~ features      = {
552         SizeFeatures        = {
553           {
554             Size            = {-8},
555             Font            = {
556               NewCM08-BookItalic.otf
557             }
558           },
559           {
560             Size            = {8-},
561             Font            = {
562               NewCM10-BookItalic.otf
563             }
564           }
565         }
566       }
567     } {
568       \lngx_set_keys:n {
569         text~
570         italic~ features    = {
571           SizeFeatures      = {
572             {
573               Size          = {-8},
574               Font          = {
575                 NewCM08-Italic.otf
576               }
577             },
578             {
579               Size          = {8-},
580               Font          = {
581                 NewCM08-Italic.otf
582               }
583             }
584           }
585         }
586       }
587     }
588   }
589 }
590 \tl_if_in:cnT {
591   g _ lngx _ text _ italic _ tl
592 } { NewCMSans } {

```

```

593 \tl_if_in:cnTF {
594   g _ lngx _ text _ italic _ tl
595 } { Book } {
596   \lngx_set_keys:n {
597     text~
598     italic~ features      = {
599       SizeFeatures      = {
600         {
601           Size           = {-8},
602           Font           = {
603             NewCMSans08-BookOblique.otf
604           }
605         },
606         {
607           Size           = {8-},
608           Font           = {
609             NewCMSans10-BookOblique.otf
610           }
611         }
612       }
613     }
614   }
615 } {
616   \lngx_set_keys:n {
617     text~
618     italic~ features      = {
619       SizeFeatures      = {
620         {
621           Size           = {-8},
622           Font           = {
623             NewCMSans08-Oblique.otf
624           }
625         },
626         {
627           Size           = {8-},
628           Font           = {
629             NewCMSans08-Oblique.otf
630           }
631         }
632       }
633     }
634   }
635 }
636 }

```

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.

```

637 \tl_if_in:cnT {
638   g _ lngx _ text _ sans _ upright _ tl
639 } { NewCMSans } {
640   \tl_if_in:cnTF {
641     g _ lngx _ text _ upright _ tl
642   } { Book } {
643     \lngx_set_keys:n {

```



```

644     text~ sans~
645     upright~ features      = {
646         SizeFeatures      = {
647             {
648                 Size        = {-8},
649                 Font        = {
650                     NewCMSans08-Book.otf
651                 }
652             },
653             {
654                 Size        = {8-},
655                 Font        = {
656                     NewCMSans10-Book.otf
657                 }
658             }
659         }
660     }
661 }
662 } {
663     \lngx_set_keys:n {
664         text~ sans~
665         upright~ features      = {
666             SizeFeatures      = {
667                 {
668                     Size        = {-8},
669                     Font        = {
670                         NewCMSans08-Regular.otf
671                     }
672                 },
673                 {
674                     Size        = {8-},
675                     Font        = {
676                         NewCMSans10-Regular.otf
677                     }
678                 }
679             }
680         }
681     }
682 }
683 }
684 \tl_if_in:cnT {
685     g _ lngx _ text _ sans _ italic _ tl
686 } { NewCMSans } {
687     \tl_if_in:cnTF {
688         g _ lngx _ text _ italic _ tl
689     } { Book } {
690         \lngx_set_keys:n {
691             text~ sans~
692             italic~ features      = {
693                 SizeFeatures      = {
694                     {
695                         Size        = {-8},
696                         Font        = {
697                             NewCMSans08-BookOblique.otf

```

```

698     }
699   },
700   {
701     Size              = {8-},
702     Font              = {
703       NewCMSans10-BookOblique.otf
704     }
705   }
706 }
707 }
708 }
709 } {
710   \lngx_set_keys:n {
711     text~ sans~
712     italic~ features = {
713       SizeFeatures   = {
714         {
715           Size        = {-8},
716           Font        = {
717             NewCMSans08-Oblique.otf
718           }
719         },
720         {
721           Size        = {8-},
722           Font        = {
723             NewCMSans10-Oblique.otf
724           }
725         }
726       }
727     }
728   }
729 }
730 }

```

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.

```

731   \lngx_set_keys:n {
732     text~
733     upright~ features = {
734       UprightFont      = {
735         \g_lngx_text_upright_tl
736       },
737       UprightFeatures  = {
738         \g_lngx_text_upright_features_tl
739       },
740       ItalicFont       = {
741         \g_lngx_text_italic_tl
742       },
743       ItalicFeatures   = {
744         \g_lngx_text_italic_features_tl
745       },
746       BoldFont         = {

```

```

747     \g_lngx_text_bold_upright_tl
748 },
749 BoldFeatures          = {
750     \g_lngx_text_bold_upright_features_tl
751 },
752 BoldItalicFont        = {
753     \g_lngx_text_bold_italic_tl
754 },
755 BoldItalicFeatures    = {
756     \g_lngx_text_bold_italic_features_tl
757 },

```

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).

```

758 \tl_if_empty:cF {
759     g _ lngx _ text _ slanted _ tl
760 } {
761     SlantedFont        = {
762         \g_lngx_text_slanted_tl
763     },
764     \tl_if_empty:cF {
765         g _ lngx _ text _ slanted _ features _ tl
766     } {
767         SlantedFeatures    = {
768             \g_lngx_text_slanted_features_tl
769         },
770     }
771 }
772 \tl_if_empty:cF {
773     g _ lngx _ text _ bold _ slanted _ tl
774 } {
775     BoldSlantedFont     = {
776         \g_lngx_text_bold_slanted_tl
777     },
778     BoldSlantedFeatures = {
779         \g_lngx_text_bold_slanted_features_tl
780     },
781 }
782 \tl_if_empty:cF {
783     g _ lngx _ text _ swash _ tl
784 } {
785     SwashFont           = {
786         \g_lngx_text_swash_tl
787     },
788     SwashFeatures       = {
789         \g_lngx_text_swash_features_tl
790     },
791 }
792 \tl_if_empty:cF {
793     g _ lngx _ text _ bold _ swash _ tl
794 } {
795     BoldSwashFont       = {
796         \g_lngx_text_bold_swash_tl
797     },

```

```

798         BoldSwashFeatures      = {
799             \g_lngx_text_bold_swash_features_tl
800         },
801     },
802     \tl_if_empty:cF {
803         g _ lngx _ text _ small _ caps _ tl
804     } {
805         SmallCapsFont           = {
806             \g_lngx_text_small_caps_tl
807         },
808         SmallCapsFeatures       = {
809             \g_lngx_text_small_caps_features_tl
810         }
811     }
812 },

```

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

```

813 text~ sans~
814 upright~ features      = {
815     UprightFont         = {
816         \g_lngx_text_sans_upright_tl
817     },
818     UprightFeatures     = {
819         \g_lngx_text_sans_upright_features_tl
820     },
821     BoldFont            = {
822         \g_lngx_text_sans_bold_upright_tl
823     },
824     BoldFeatures        = {
825         \g_lngx_text_sans_bold_upright_features_tl
826     },
827     ItalicFont          = {
828         \g_lngx_text_sans_italic_tl
829     },
830     ItalicFeatures      = {
831         \g_lngx_text_sans_italic_features_tl
832     },
833     BoldItalicFont      = {
834         \g_lngx_text_sans_bold_italic_tl
835     },
836     BoldItalicFeatures  = {
837         \g_lngx_text_sans_bold_italic_features_tl
838     },
839     \tl_if_empty:cF {
840         g _ lngx _ text _ sans _ slanted _ tl
841     } {
842         SlantedFont      = {
843             \g_lngx_text_sans_slanted_tl
844         },
845         SlantedFeatures  = {
846             \g_lngx_text_sans_slanted_features_tl
847         },
848     }
849     \tl_if_empty:cF {
850         g _ lngx _ text _ sans _ bold _ slanted _ tl

```

```

851 } {
852     BoldSlantedFont      = {
853         \g_lngx_text_sans_bold_slanted_tl
854     },
855     BoldSlantedFeatures  = {
856         \g_lngx_text_sans_bold_slanted_features_tl
857     },
858 }
859 \tl_if_empty:cF {
860     g _ lngx _ text _ sans _ swash _ tl
861 } {
862     SwashFont            = {
863         \g_lngx_text_sans_swash_tl
864     },
865     SwashFeatures        = {
866         \g_lngx_text_sans_swash_features_tl
867     },
868 }
869 \tl_if_empty:cF {
870     g _ lngx _ text _ sans _ bold _ swash _ tl
871 } {
872     BoldSwashFont        = {
873         \g_lngx_text_sans_bold_swash_tl
874     },
875     BoldSwashFeatures    = {
876         \g_lngx_text_sans_bold_swash_features_tl
877     },
878 }
879 \tl_if_empty:cF {
880     g _ lngx _ text _ sans _ small _ caps _ tl
881 } {
882     SmallCapsFont        = {
883         \g_lngx_text_sans_small_caps_tl
884     },
885     SmallCapsFeatures    = {
886         \g_lngx_text_sans_small_caps_features_tl
887     }
888 }
889 },
890 text~ mono~
891 upright~ features      = {
892     UprightFont         = {
893         \g_lngx_text_mono_upright_tl
894     },
895     UprightFeatures     = {
896         \g_lngx_text_mono_upright_features_tl
897     },
898     BoldFont            = {
899         \g_lngx_text_mono_bold_upright_tl
900     },
901     BoldFeatures        = {
902         \g_lngx_text_mono_bold_upright_features_tl
903     },
904     ItalicFont          = {

```

```

905     \g_lngx_text_mono_italic_tl
906 },
907 ItalicFeatures          = {
908     \g_lngx_text_mono_italic_features_tl
909 },
910 BoldItalicFont          = {
911     \g_lngx_text_mono_bold_italic_tl
912 },
913 BoldItalicFeatures      = {
914     \g_lngx_text_mono_bold_italic_features_tl
915 },
916 \tl_if_empty:cF {
917     g _ lngx _ text _ mono _ slanted _ tl
918 } {
919     SlantedFont          = {
920         \g_lngx_text_mono_slanted_tl
921     },
922     SlantedFeatures      = {
923         \g_lngx_text_mono_slanted_features_tl
924     },
925 }
926 \tl_if_empty:cF {
927     g _ lngx _ text _ mono _ bold _ slanted _ tl
928 } {
929     BoldSlantedFont      = {
930         \g_lngx_text_mono_bold_slanted_tl
931     },
932     BoldSlantedFeatures  = {
933         \g_lngx_text_mono_bold_slanted_features_tl
934     },
935 }
936 \tl_if_empty:cF {
937     g _ lngx _ text _ mono _ swash _ tl
938 } {
939     SwashFont            = {
940         \g_lngx_text_mono_swash_tl
941     },
942     SwashFeatures        = {
943         \g_lngx_text_mono_swash_features_tl
944     },
945 }
946 \tl_if_empty:cF {
947     g _ lngx _ text _ mono _ bold _ swash _ tl
948 } {
949     BoldSwashFont        = {
950         \g_lngx_text_mono_bold_swash_tl
951     },
952     BoldSwashFeatures    = {
953         \g_lngx_text_mono_bold_swash_features_tl
954     },
955 }
956 \tl_if_empty:cF {
957     g _ lngx _ text _ mono _ small _ caps _ tl
958 } {

```

```

959     SmallCapsFont          = {
960         \g_lngx_text_mono_small_caps_tl
961     },
962     SmallCapsFeatures      = {
963         \g_lngx_text_mono_small_caps_features_tl
964     }
965 }
966 }
967 }
968 \bool_if:NT \g_lngx_old_style_bool {
969     \lngx_set_keys:n {
970         text~
971         upright~ features      = {
972             Numbers            = { OldStyle }
973         },
974         text~ sans~
975         upright~ features      = {
976             Numbers            = { OldStyle }
977         }
978     }
979     \tl_if_in:cnT {
980         g_lngx_math_tl
981     } { NewCM } {
982         \bool_if:NT \g_lngx_old_style_one_bool {
983             \lngx_set_keys:n {
984                 text~
985                 upright~ features      = {
986                     CharacterVariant    = { 6 }
987                 },
988                 text~ sans~
989                 upright~ features      = {
990                     CharacterVariant    = { 6 }
991                 }
992             }
993         }
994     }
995 }
996 \tl_if_in:cnT {
997     g _ lngx _ math _ tl
998 } { NewCM } {
999     \bool_if:NT \g_lngx_bourbaki_bool {
1000         \lngx_set_keys:n {
1001             math~ features            = {
1002                 CharacterVariant      = { 1 }
1003             }
1004         }
1005     }
1006 }

```

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.

```

1007 \tl_if_in:cnT {
1008     g _ lngx _ text _ upright _ tl

```

```

1009 } { NewCM } {
1010   \lngx_set_keys:n {
1011     text~
1012     upright~ features          = {
1013       IgnoreFontspecFile
1014     }
1015   }
1016 }
1017 \tl_if_in:cnT {
1018   g _ lngx _ text _ sans _ upright _ tl
1019 } { NewCM } {
1020   \lngx_set_keys:n {
1021     text~
1022     sans~ upright~ features    = {
1023       IgnoreFontspecFile
1024     }
1025   }
1026 }
1027 \tl_if_in:cnT {
1028   g _ lngx _ text _ mono _ upright _ tl
1029 } { NewCM } {
1030   \lngx_set_keys:n {
1031     text~
1032     mono~ upright~ features    = {
1033       IgnoreFontspecFile
1034     }
1035   }
1036 }
1037 \lngx_set_main_font:ee {
1038   \g_lngx_text_upright_features_tl
1039 } {
1040   \g_lngx_text_upright_tl
1041 }
1042 \lngx_set_sans_font:ee {
1043   \g_lngx_text_sans_upright_features_tl
1044 } {
1045   \g_lngx_text_sans_upright_tl
1046 }
1047 \lngx_set_mono_font:ee {
1048   \g_lngx_text_mono_upright_features_tl
1049 } {
1050   \g_lngx_text_mono_upright_tl
1051 }
1052 \lngx_set_math_font:ee {
1053   \g_lngx_math_features_tl
1054 } {
1055   \g_lngx_math_tl
1056 }
1057 }
1058 </font>

```



```

1059 <*ipa>
1060 \ProvidesExplPackage{linguistix-ipa}
1061     {2025-05-20}
1062     {v0.1b}
1063     {%
1064         A package for typesetting the IPA
1065         (International Phonetic Alphabet) from
1066         the ‘Linguistix’ bundle.%
1067     }

```

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

```

1068
1069 \IfPackageLoadedF { unicode-math } {
1070     \RequirePackage { unicode-math }
1071 }
1072
1073 \IfPackageLoadedF { linguistix-base } {
1074     \RequirePackage { linguistix-base }
1075 }
1076
1077 \IfPackageLoadedF { linguistix-nfss } {
1078     \RequirePackage { linguistix-nfss }
1079 }

```

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

```

1080
1081 \NewDocumentCommand \ipatext { s m } {
1082     \IfBooleanTF { #1 } {
1083         {
1084             \lngxipa
1085             / #2 /
1086         }
1087     } {
1088         {
1089             \lngxipa
1090             [ #2 ]
1091         }
1092     }
1093 }

```

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

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

```

```

1094
1095 \clist_map_inline:nn {
1096   upright,
1097   bold~ upright,
1098   italic,
1099   bold~ italic,
1100   slanted,
1101   bold~ slanted,
1102   swash,
1103   bold~ swash,
1104   small~ caps
1105 } {
1106   \tl_set:Nn \l_tmpa_tl { #1 }
1107   \tl_replace_all:Nnn \l_tmpa_tl { ~ } { _ }
1108   \tl_gclear_new:c {
1109     g _ lngx _ ipa _ \l_tmpa_tl _ features _ tl
1110   }
1111   \keys_define:nn { lngx _ keys } {
1112     ipa~ #1
1113     .tl_gset_e:c          = {
1114       g _ lngx _ ipa _ \l_tmpa_tl _ tl
1115     },
1116     ipa~ #1~ features
1117     .code:n              = {
1118       \tl_set:Nn \l_tmpb_tl { #1 }
1119       \tl_replace_all:Nnn \l_tmpb_tl { ~ } { _ }
1120       \tl_put_right:ce {
1121         g _ lngx _ ipa _ \l_tmpb_tl _ features _ tl
1122       } { ##1 , }
1123       \tl_clear:N \l_tmpb_tl
1124     }
1125   }
1126   \tl_clear:N \l_tmpa_tl
1127 }

```

(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

```

```

II28
II29 \clist_map_inline:nn {
II30   upright,
II31   bold~ upright,
II32   italic,
II33   bold~ italic,
II34   slanted,
II35   bold~ slanted,
II36   swash,
II37   bold~ swash,
II38   small~ caps
II39 } {
II40   \tl_set:Nn \l_tmpa_tl { #1 }
II41   \tl_replace_all:Nnn \l_tmpa_tl { ~ } { _ }
II42   \tl_gclear_new:c {
II43     g _ lngx _ ipa _ mono _ \l_tmpa_tl _ features _ tl
II44   }
II45   \keys_define:nn { lngx _ keys } {
II46     ipa~ mono~ #1
II47     .tl_gset_e:c          = {
II48       g _ lngx _ ipa _ mono _ \l_tmpa_tl _ tl
II49     },
II50     ipa~ mono~ #1~ features
II51     .code:n              = {
II52       \tl_set:Nn \l_tmpb_tl { #1 }
II53       \tl_replace_all:Nnn \l_tmpb_tl { ~ } { _ }
II54       \tl_put_right:ce {
II55         g _ lngx _ ipa _ mono _ \l_tmpb_tl _ features _ tl
II56       } { ##1 , }
II57       \tl_clear:N \l_tmpb_tl
II58     }
II59   }
II60   \tl_clear:N \l_tmpa_tl
II61 }

```

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

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

```

```

ii62
ii63 \clist_map_inline:nn {
ii64   upright,
ii65   bold~ upright,
ii66   italic,
ii67   bold~ italic,
ii68   slanted,
ii69   bold~ slanted,
ii70   swash,
ii71   bold~ swash,
ii72   small~ caps
ii73 } {
ii74   \tl_set:Nn \l_tmpa_tl { #1 }
ii75   \tl_replace_all:Nnn \l_tmpa_tl { ~ } { _ }
ii76   \tl_gclear_new:c {
ii77     g _ lngx _ ipa _ sans _ \l_tmpa_tl _ features _ tl
ii78   }
ii79   \keys_define:nn { lngx _ keys } {
ii80     ipa~ sans~ #1
ii81     .tl_gset_e:c          = {
ii82       g _ lngx _ ipa _ sans _ \l_tmpa_tl _ tl
ii83     },
ii84     ipa~ sans~ #1~ features
ii85     .code:n              = {
ii86       \tl_set:Nn \l_tmpb_tl { #1 }
ii87       \tl_replace_all:Nnn \l_tmpb_tl { ~ } { _ }
ii88       \tl_put_right:ce {
ii89         g _ lngx _ ipa _ sans _ \l_tmpb_tl _ features _ tl
ii90       } { ##1 , }
ii91       \tl_clear:N \l_tmpb_tl
ii92     }
ii93   }
ii94   \tl_clear:N \l_tmpa_tl
ii95 }

```

(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.

```

ii96
ii97 \keys_define:nn { lngx _ keys } {
ii98   ipa~ newcm
ii99   .meta:n          = {
i200     ipa~
i201     upright        = {
i202       NewCM10-Book.otf
i203     },
i204     ipa~
i205     bold~ upright   = {
i206       NewCM10-Bold.otf
i207     },
i208     ipa~
i209     italic          = {

```

```

I210     NewCM10-BookItalic.otf
I211 },
I212 ipa~
I213 bold~ italic          = {
I214     NewCM10-BoldItalic.otf
I215 },
I216 ipa~
I217 slanted               = {
I218     NewCM10-Book.otf
I219 },
I220 ipa~
I221 bold~ slanted         = {
I222     NewCM10-Bold.otf
I223 },
I224 ipa~
I225 swash                 = {
I226     NewCM10-Book.otf
I227 },
I228 ipa~
I229 bold~ swash           = {
I230     NewCM10-Bold.otf
I231 },
I232 ipa~
I233 small~ caps           = {
I234     NewCM10-Book.otf
I235 },
I236 ipa~
I237 sans~ upright         = {
I238     NewCMSans10-Book.otf
I239 },
I240 ipa~
I241 sans~ bold~ upright   = {
I242     NewCMSans10-Bold.otf
I243 },
I244 ipa~
I245 sans~ italic          = {
I246     NewCMSans10-BookOblique.otf
I247 },
I248 ipa~
I249 sans~ bold~ italic    = {
I250     NewCMSans10-BoldOblique.otf
I251 },
I252 ipa~
I253 sans~ slanted         = {
I254     NewCMSans10-BookOblique.otf
I255 },
I256 ipa~
I257 sans~ bold~ slanted   = {
I258     NewCMSans10-BoldOblique.otf
I259 },
I260 ipa~
I261 sans~ swash           = {
I262     NewCMSans10-Book.otf
I263 },

```

```

1264   ipa~
1265   sans~ bold~ swash      = {
1266     NewCMSans10-Bold.otf
1267   },
1268   ipa~
1269   sans~ small~ caps      = {
1270     NewCMSans10-Book.otf
1271   },
1272   ipa~
1273   mono~ upright          = {
1274     NewCMMono10-Book.otf
1275   },
1276   ipa~
1277   mono~ bold~ upright    = {
1278     NewCMMono10-Bold.otf
1279   },
1280   ipa~
1281   mono~ italic           = {
1282     NewCMMono10-BookItalic.otf
1283   },
1284   ipa~
1285   mono~ bold~ italic     = {
1286     NewCMMono10-BoldOblique.otf
1287   },
1288   ipa~
1289   mono~ slanted          = {
1290     NewCMMono10-Book.otf
1291   },
1292   ipa~
1293   mono~ bold~ slanted    = {
1294     NewCMMono10-BoldOblique.otf
1295   },
1296   ipa~
1297   mono~ swash            = {
1298     NewCMMono10-Book.otf
1299   },
1300   ipa~
1301   mono~ bold~ swash      = {
1302     NewCMMono10-Bold.otf
1303   },
1304   ipa~
1305   mono~ small~ caps      = {
1306     NewCMMono10-Book.otf
1307   }
1308 }
1309 }

```

(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.

```

1310
1311 \keys_define:nn { lngx _ keys } {
1312   ipa~ newcm~ sans

```

```

1313 .meta:n          = {
1314   ipa~
1315   upright          = {
1316     NewCMSans10-Book.otf
1317   },
1318   ipa~
1319   bold~ upright    = {
1320     NewCMSans10-Bold.otf
1321   },
1322   ipa~
1323   italic           = {
1324     NewCMSans10-BookOblique.otf
1325   },
1326   ipa~
1327   bold~ italic     = {
1328     NewCMSans10-BoldOblique.otf
1329   },
1330   ipa~
1331   slanted          = {
1332     NewCMSans10-BookOblique.otf
1333   },
1334   ipa~
1335   bold~ slanted    = {
1336     NewCMSans10-BoldOblique.otf
1337   },
1338   ipa~
1339   swash            = {
1340     NewCMSans10-Book.otf
1341   },
1342   ipa~
1343   bold~ swash      = {
1344     NewCMSans10-Bold.otf
1345   },
1346   ipa~
1347   small~ caps      = {
1348     NewCMSans10-Book.otf
1349   }
1350 }
1351 }

```

(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.

```

1352
1353 \keys_define:nn { lngx _ keys } {
1354   ipa~ newcm~ mono
1355   .meta:n          = {
1356     ipa~
1357     upright          = {
1358       NewCMMono10-Book.otf
1359     },
1360     ipa~
1361     bold~ upright    = {

```

```

1362     NewCMMono10-Bold.otf
1363 },
1364 ipa~
1365 italic                = {
1366     NewCMMono10-BookItalic.otf
1367 },
1368 ipa~
1369 bold~ italic          = {
1370     NewCMMono10-BoldOblique.otf
1371 },
1372 ipa~
1373 slanted                = {
1374     NewCMMono10-Book.otf
1375 },
1376 ipa~
1377 bold~ slanted          = {
1378     NewCMMono10-BoldOblique.otf
1379 },
1380 ipa~
1381 swash                  = {
1382     NewCMMono10-Book.otf
1383 },
1384 ipa~
1385 bold~ swash            = {
1386     NewCMMono10-Bold.otf
1387 },
1388 ipa~
1389 small~ caps            = {
1390     NewCMMono10-Book.otf
1391 }
1392 }
1393 }

```

(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.

```

1394
1395 \keys_define:nn { lngx _ keys } {
1396   ipa~ newcm~ regular
1397   .meta:n                = {
1398     ipa~
1399     upright                = {
1400         NewCM10-Regular.otf
1401     },
1402     ipa~
1403     bold~ upright          = {
1404         NewCM10-Bold.otf
1405     },
1406     ipa~
1407     italic                 = {
1408         NewCM10-Italic.otf
1409     },
1410     ipa~

```



```

I411 bold~ italic = {
I412     NewCM10-BoldItalic.otf
I413 },
I414 ipa~
I415 slanted = {
I416     NewCM10-Regular.otf
I417 },
I418 ipa~
I419 bold~ slanted = {
I420     NewCM10-Bold.otf
I421 },
I422 ipa~
I423 swash = {
I424     NewCM10-Regular.otf
I425 },
I426 ipa~
I427 bold~ swash = {
I428     NewCM10-Bold.otf
I429 },
I430 ipa~
I431 small~ caps = {
I432     NewCM10-Regular.otf
I433 },
I434 ipa~
I435 sans~ upright = {
I436     NewCMSans10-Regular.otf
I437 },
I438 ipa~
I439 sans~ bold = {
I440     NewCMSans10-Bold.otf
I441 },
I442 ipa~
I443 sans~ italic = {
I444     NewCMSans10-Oblique.otf
I445 },
I446 ipa~
I447 sans~ bold~ italic = {
I448     NewCMSans10-BoldOblique.otf
I449 },
I450 ipa~
I451 sans~ slanted = {
I452     NewCMSans10-Regular.otf
I453 },
I454 ipa~
I455 sans~ bold~ slanted = {
I456     NewCMSans10-Bold.otf
I457 },
I458 ipa~
I459 sans~ swash = {
I460     NewCMSans10-Regular.otf
I461 },
I462 ipa~
I463 sans~ bold~ swash = {
I464     NewCMSans10-Bold.otf

```

```

1465 },
1466 ipa~
1467 sans~ small~ caps      = {
1468     NewCMSans10-Regular.otf
1469 },
1470 ipa~
1471 mono~ upright          = {
1472     NewCMMono10-Regular.otf
1473 },
1474 ipa~
1475 mono~ bold             = {
1476     NewCMMono10-Bold.otf
1477 },
1478 ipa~
1479 mono~ italic           = {
1480     NewCMMono10-Italic.otf
1481 },
1482 ipa~
1483 mono~ bold~ italic     = {
1484     NewCMMono10-Bold.otf
1485 },
1486 ipa~
1487 mono~ slanted          = {
1488     NewCMMono10-Regular.otf
1489 },
1490 ipa~
1491 mono~ bold~ slanted    = {
1492     NewCMMono10-Bold.otf
1493 },
1494 ipa~
1495 mono~ swash            = {
1496     NewCMMono10-Regular.otf
1497 },
1498 ipa~
1499 mono~ bold~ swash      = {
1500     NewCMMono10-Bold.otf
1501 },
1502 ipa~
1503 mono~ small~ caps      = {
1504     NewCMMono10-Regular.otf
1505 }
1506 }
1507 }

```

(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.

```

1508
1509 \keys_define:nn { lngx _ keys } {
1510     ipa~ newcm~ sans~ regular
1511     .meta:n          = {
1512         ipa~
1513         upright          = {

```

```

1514     NewCMSans10-Regular.otf
1515 },
1516 ipa~
1517 bold = {
1518     NewCMSans10-Bold.otf
1519 },
1520 ipa~
1521 italic = {
1522     NewCMSans10-Oblique.otf
1523 },
1524 ipa~
1525 bold~ italic = {
1526     NewCMSans10-BoldOblique.otf
1527 },
1528 ipa~
1529 slanted = {
1530     NewCMSans10-Regular.otf
1531 },
1532 ipa~
1533 bold~ slanted = {
1534     NewCMSans10-Bold.otf
1535 },
1536 ipa~
1537 swash = {
1538     NewCMSans10-Regular.otf
1539 },
1540 ipa~
1541 bold~ swash = {
1542     NewCMSans10-Bold.otf
1543 },
1544 ipa~
1545 small~ caps = {
1546     NewCMSans10-Regular.otf
1547 }
1548 }
1549 }

```

(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.

```

1550
1551 \keys_define:nn { lngx _ keys } {
1552   ipa~ newcm~ mono~ regular
1553   .meta:n = {
1554     ipa~
1555     upright = {
1556         NewCMMono10-Regular.otf
1557     },
1558     ipa~
1559     bold = {
1560         NewCMMono10-Bold.otf
1561     },
1562     ipa~

```

```

1563     italic                = {
1564         NewCMMono10-Italic.otf
1565     },
1566     ipa~
1567     bold~ italic          = {
1568         NewCMMono10-Bold.otf
1569     },
1570     ipa~
1571     slanted                = {
1572         NewCMMono10-Regular.otf
1573     },
1574     ipa~
1575     bold~ slanted          = {
1576         NewCMMono10-Bold.otf
1577     },
1578     ipa~
1579     swash                  = {
1580         NewCMMono10-Regular.otf
1581     },
1582     ipa~
1583     bold~ swash            = {
1584         NewCMMono10-Bold.otf
1585     },
1586     ipa~
1587     small~ caps            = {
1588         NewCMMono10-Regular.otf
1589     }
1590 }
1591 }

```

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

We set the `ipa newcm` key by default.

```

1592
1593 \ltx_set_keys:n {ipa~ newcm}

```

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

```

1594
1595 \sys_if_engine luatex:T {
1596     \ltx_set_keys:n {
1597         ipa~
1598         upright~ features    = {
1599             Renderer          = { HarfBuzz }
1600         },
1601         ipa~ sans~
1602         upright~ features    = {
1603             Renderer          = { HarfBuzz }
1604         },
1605         ipa~ mono~
1606         upright~ features    = {
1607             Renderer          = { HarfBuzz }
1608         }
1609     }
1610 }

```

`\lngx_set_main_ipa_font:nn` 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.

```

\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
r611
r612 \cs_new_protected:Npn \lngx_set_main_ipa_font:nn #1#2 {
r613   \setfontfamily \lngx_main_ipa: [
r614     #1,
r615     NFSSFamily           = { lngx _ ipa _ rm _ nfss }
r616   ] { #2 }
r617 }
r618
r619 \cs_new_protected:Npn \lngx_set_sans_ipa_font:nn #1#2 {
r620   \setfontfamily \lngx_sans_ipa: [
r621     #1,
r622     NFSSFamily           = { lngx _ ipa _ sf _ nfss }
r623   ] { #2 }
r624 }
r625
r626 \cs_new_protected:Npn \lngx_set_mono_ipa_font:nn #1#2 {
r627   \setfontfamily \lngx_mono_ipa: [
r628     #1,
r629     NFSSFamily           = { lngx _ ipa _ tt _ nfss }
r630   ] { #2 }
r631 }
r632
r633 \cs_generate_variant:Nn \lngx_set_main_ipa_font:nn { ee }
r634 \cs_generate_variant:Nn \lngx_set_sans_ipa_font:nn { ee }
r635 \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 12.)

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

```

r636
r637 \lngx_super_font_family:nn { lngx _ ipa } {
r638   rm           = { lngx _ ipa _ rm _ nfss },
r639   sf           = { lngx _ ipa _ sf _ nfss },
r640   tt           = { lngx _ ipa _ tt _ nfss }
r641 }

```

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

\lngxipa I use `\lngx softer_super_font_family:n` provided by LINGUIS~~TI~~X-NFSS for defining this switch to the IPA.

```

r642
r643 \cs_new_protected:Npn \lngx_ipa: {
r644   \lngx softer_super_font_family:n { lngx _ ipa }
r645 }
r646
r647 \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}`.

```

1648
1649 \hook_gput_code:nnn { begindocument / before } { . } {
1650   \tl_if_in:cnT {
1651     g _ lngx _ ipa _ upright _ tl
1652   } { NewCM } {
1653     \tl_if_in:cnTF {
1654       g _ lngx _ ipa _ upright _ tl
1655     } { Book } {
1656       \lngx_set_keys:n {
1657         ipa~
1658         upright~ features      = {
1659           SizeFeatures         = {
1660             {
1661               Size              = {-8},
1662               Font              = {
1663                 NewCM08-Book.otf
1664             }
1665           },
1666           {
1667             Size                = {8-},
1668             Font                = {
1669               NewCM10-Book.otf
1670             }
1671           }
1672         }
1673       }
1674     }
1675   } {
1676     \lngx_set_keys:n {
1677       ipa~
1678       upright~ features      = {
1679         SizeFeatures         = {
1680           {
1681             Size                = {-8},
1682             Font                = {
1683               NewCM08-Regular.otf
1684             }
1685           },
1686           {
1687             Size                = {8-},
1688             Font                = {
1689               NewCM10-Regular.otf
1690             }
1691           }
1692         }
1693       }
1694     }
1695   }
1696 }
1697 \tl_if_in:cnT {

```

```

1698     g _ lngx _ ipa _ upright _ tl
1699 } { NewCMSans } {
1700     \tl_if_in:cnTF {
1701         g _ lngx _ ipa _ upright _ tl
1702     } { Book } {
1703         \lngx_set_keys:n {
1704             ipa~
1705             upright~ features      = {
1706                 SizeFeatures      = {
1707                     {
1708                         Size        = {-8},
1709                         Font        = {
1710                             NewCMSans08-Book.otf
1711                         }
1712                     },
1713                     {
1714                         Size        = {8-},
1715                         Font        = {
1716                             NewCMSans10-Book.otf
1717                         }
1718                     }
1719                 }
1720             }
1721         } {
1722             \lngx_set_keys:n {
1723                 ipa~
1724                 upright~ features  = {
1725                     SizeFeatures  = {
1726                         {
1727                             Size        = {-8},
1728                             Font        = {
1729                                 NewCMSans08-Regular.otf
1730                             }
1731                         },
1732                         {
1733                             Size        = {8-},
1734                             Font        = {
1735                                 NewCMSans10-Regular.otf
1736                             }
1737                         }
1738                     }
1739                 }
1740             }
1741         }
1742     }
1743 }
1744 \tl_if_in:cnT {
1745     g _ lngx _ ipa _ italic _ tl
1746 } { NewCM } {
1747     \tl_if_in:cnTF {
1748         g _ lngx _ ipa _ italic _ tl
1749     } { Book } {
1750         \lngx_set_keys:n {
1751             ipa~

```

```

1752         italic~ features          = {
1753             SizeFeatures           = {
1754                 {
1755                     Size             = {-8},
1756                     Font             = {
1757                         NewCM08-BookItalic.otf
1758                     }
1759                 },
1760                 {
1761                     Size             = {8-},
1762                     Font             = {
1763                         NewCM10-BookItalic.otf
1764                     }
1765                 }
1766             }
1767         }
1768     }
1769 } {
1770     \lngx_set_keys:n {
1771         ipa~
1772         italic~ features          = {
1773             SizeFeatures           = {
1774                 {
1775                     Size             = {-8},
1776                     Font             = {
1777                         NewCM08-Italic.otf
1778                     }
1779                 },
1780                 {
1781                     Size             = {8-},
1782                     Font             = {
1783                         NewCM08-Italic.otf
1784                     }
1785                 }
1786             }
1787         }
1788     }
1789 }
1790 }
1791 \tl_if_in:cnT {
1792     g _ lngx _ ipa _ italic _ tl
1793 } { NewCMSans } {
1794     \tl_if_in:cnTF {
1795         g _ lngx _ ipa _ italic _ tl
1796     } { Book } {
1797         \lngx_set_keys:n {
1798             ipa~
1799             italic~ features          = {
1800                 SizeFeatures           = {
1801                     {
1802                         Size             = {-8},
1803                         Font             = {
1804                             NewCMSans08-BookOblique.otf
1805                     }

```



```

1806         },
1807         {
1808             Size              = {-8},
1809             Font              = {
1810                 NewCMSans10-BookOblique.otf
1811             }
1812         }
1813     }
1814 }
1815 }
1816 } {
1817     \lngx_set_keys:n {
1818         ipa~
1819         italic~ features      = {
1820             SizeFeatures      = {
1821                 {
1822                     Size              = {-8},
1823                     Font              = {
1824                         NewCMSans08-Oblique.otf
1825                     }
1826                 },
1827                 {
1828                     Size              = {-8},
1829                     Font              = {
1830                         NewCMSans08-Oblique.otf
1831                     }
1832                 }
1833             }
1834         }
1835     }
1836 }
1837 }
1838 \tl_if_in:cnT {
1839     g _ lngx _ ipa _ sans _ upright _ tl
1840 } { NewCMSans } {
1841     \tl_if_in:cnTF {
1842         g _ lngx _ ipa _ upright _ tl
1843     } { Book } {
1844         \lngx_set_keys:n {
1845             ipa~ sans~
1846             upright~ features      = {
1847                 SizeFeatures      = {
1848                     {
1849                         Size              = {-8},
1850                         Font              = {
1851                             NewCMSans08-Book.otf
1852                         }
1853                     },
1854                     {
1855                         Size              = {-8},
1856                         Font              = {
1857                             NewCMSans10-Book.otf
1858                         }
1859                     }
1860                 }
1861             }
1862         }
1863     }
1864 }

```

```

1860     }
1861   }
1862 }
1863 } {
1864   \ltx_set_keys:n {
1865     ipa~ sans~
1866     upright~ features      = {
1867       SizeFeatures        = {
1868         {
1869           Size              = {-8},
1870           Font              = {
1871             NewCMSans08-Regular.otf
1872           }
1873         },
1874         {
1875           Size              = {8-},
1876           Font              = {
1877             NewCMSans10-Regular.otf
1878           }
1879         }
1880       }
1881     }
1882   }
1883 }
1884 }
1885 \tl_if_in:cnT {
1886   g _ lngx _ ipa _ sans _ italic _ tl
1887 } { NewCMSans } {
1888   \tl_if_in:cnTF {
1889     g _ lngx _ ipa _ italic _ tl
1890   } { Book } {
1891     \ltx_set_keys:n {
1892       ipa~ sans~
1893       italic~ features      = {
1894         SizeFeatures        = {
1895           {
1896             Size              = {-8},
1897             Font              = {
1898               NewCMSans08-BookOblique.otf
1899             }
1900           },
1901           {
1902             Size              = {8-},
1903             Font              = {
1904               NewCMSans10-BookOblique.otf
1905             }
1906           }
1907         }
1908       }
1909     }
1910   } {
1911     \ltx_set_keys:n {
1912       ipa~ sans~
1913       italic~ features      = {

```

```

1914         SizeFeatures          = {
1915             {
1916                 Size            = {-8},
1917                 Font            = {
1918                     NewCMSans08-Oblique.otf
1919                 }
1920             },
1921             {
1922                 Size            = {8-},
1923                 Font            = {
1924                     NewCMSans10-Oblique.otf
1925                 }
1926             }
1927         }
1928     }
1929 }
1930 }
1931 }

```

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

```

1932 \lngx_set_keys:n {
1933     ipa~ upright~ features = {
1934         UprightFont        = {
1935             \g_lngx_ipa_upright_tl
1936         },
1937         UprightFeatures     = {
1938             \g_lngx_ipa_upright_features_tl
1939         },
1940         BoldFont            = {
1941             \g_lngx_ipa_bold_upright_tl
1942         },
1943         BoldFeatures        = {
1944             \g_lngx_ipa_bold_upright_features_tl
1945         },
1946         ItalicFont          = {
1947             \g_lngx_ipa_italic_tl
1948         },
1949         ItalicFeatures      = {
1950             \g_lngx_ipa_italic_features_tl
1951         },
1952         BoldItalicFont      = {
1953             \g_lngx_ipa_bold_italic_tl
1954         },
1955         BoldItalicFeatures  = {
1956             \g_lngx_ipa_bold_italic_features_tl
1957         },
1958         \tl_if_empty:cF {
1959             g _ lngx _ ipa _ slanted _ tl
1960         } {
1961             SlantedFont      = {
1962                 \g_lngx_ipa_slanted_tl
1963             },
1964             \tl_if_empty:cF {
1965                 g _ lngx _ ipa _ slanted _ features _ tl
1966             } {

```

```

1967         SlantedFeatures           = {
1968             \g_lngx_ipa_slanted_features_tl
1969         },
1970     }
1971 }
1972 \tl_if_empty:cF {
1973     g _ lngx _ ipa _ bold _ slanted _ tl
1974 } {
1975     BoldSlantedFont           = {
1976         \g_lngx_ipa_bold_slanted_tl
1977     },
1978     \tl_if_empty:cF {
1979         g _ lngx _ ipa _ bold _ slanted _ features _ tl
1980     } {
1981         BoldSlantedFeatures    = {
1982             \g_lngx_ipa_bold_slanted_features_tl
1983         },
1984     }
1985 }
1986 \tl_if_empty:cF {
1987     g _ lngx _ ipa _ swash _ tl
1988 } {
1989     SwashFont                 = {
1990         \g_lngx_ipa_swash_tl
1991     },
1992     \tl_if_empty:cF {
1993         g _ lngx _ ipa _ swash _ features _ tl
1994     } {
1995         SwashFeatures          = {
1996             \g_lngx_ipa_swash_features_tl
1997         },
1998     }
1999 }
2000 \tl_if_empty:cF {
2001     g _ lngx _ ipa _ bold _ swash _ tl
2002 } {
2003     BoldSwashFont             = {
2004         \g_lngx_ipa_bold_swash_tl
2005     },
2006     \tl_if_empty:cF {
2007         g _ lngx _ ipa _ bold _ swash _ features _ tl
2008     } {
2009         BoldSwashFeatures      = {
2010             \g_lngx_ipa_bold_swash_features_tl
2011         },
2012     }
2013 }
2014 \tl_if_empty:cF {
2015     g _ lngx _ ipa _ small _ caps _ tl
2016 } {
2017     SmallCapsFont             = {
2018         \g_lngx_ipa_small_caps_tl
2019     }
2020     \tl_if_empty:cF {

```

```

2021         g _ lngx _ ipa _ small _ caps _ features _ tl
2022     } {
2023         SmallCapsFeatures          = {
2024             \g_lngx_ipa_small_caps_features_tl
2025         }
2026     }
2027 }
2028 },
2029 ipa~
2030 sans~ upright~ features = {
2031     UprightFont                = {
2032         \g_lngx_ipa_sans_upright_tl
2033     },
2034     UprightFeatures            = {
2035         \g_lngx_ipa_sans_upright_features_tl
2036     },
2037     BoldFont                   = {
2038         \g_lngx_ipa_sans_bold_upright_tl
2039     },
2040     BoldFeatures               = {
2041         \g_lngx_ipa_sans_bold_upright_features_tl
2042     },
2043     ItalicFont                 = {
2044         \g_lngx_ipa_sans_italic_tl
2045     },
2046     ItalicFeatures             = {
2047         \g_lngx_ipa_sans_italic_features_tl
2048     },
2049     BoldItalicFont             = {
2050         \g_lngx_ipa_sans_bold_italic_tl
2051     },
2052     BoldItalicFeatures         = {
2053         \g_lngx_ipa_sans_bold_italic_features_tl
2054     },
2055     \tl_if_empty:cF {
2056         g _ lngx _ ipa _ slanted _ tl
2057     } {
2058         SlantedFont            = {
2059             \g_lngx_ipa_slanted_tl
2060         },
2061         \tl_if_empty:cF {
2062             g _ lngx _ ipa _ slanted _ features _ tl
2063         } {
2064             SlantedFeatures    = {
2065                 \g_lngx_ipa_slanted_features_tl
2066             },
2067         }
2068     }
2069     \tl_if_empty:cF {
2070         g _ lngx _ ipa _ sans _ bold _ slanted _ tl
2071     } {
2072         BoldSlantedFont        = {
2073             \g_lngx_ipa_sans_bold_slanted_tl
2074         },

```

```

2075     \tl_if_empty:cF {
2076         g _ lngx _ ipa _ sans _ bold _ slanted _ features
2077         _ tl
2078     } {
2079         BoldSlantedFeatures = {
2080             \g_lngx_ipa_sans_bold_slanted_features_tl
2081         },
2082     }
2083 }
2084 \tl_if_empty:cF {
2085     g _ lngx _ ipa _ sans _ swash _ tl
2086 } {
2087     SwashFont = {
2088         \g_lngx_ipa_sans_swash_tl
2089     },
2090     \tl_if_empty:cF {
2091         g _ lngx _ ipa _ sans _ swash _ features _ tl
2092     } {
2093         SwashFeatures = {
2094             \g_lngx_ipa_sans_swash_features_tl
2095         },
2096     }
2097 }
2098 \tl_if_empty:cF {
2099     g _ lngx _ ipa _ sans _ bold _ swash _ tl
2100 } {
2101     BoldSwashFont = {
2102         \g_lngx_ipa_sans_bold_swash_tl
2103     },
2104     \tl_if_empty:cF {
2105         g _ lngx _ ipa _ sans _ bold _ swash _ features _
2106         tl
2107     } {
2108         BoldSwashFeatures = {
2109             \g_lngx_ipa_sans_bold_swash_features_tl
2110         },
2111     }
2112 }
2113 \tl_if_empty:cF {
2114     g _ lngx _ ipa _ sans _ small _ caps _ tl
2115 } {
2116     SmallCapsFont = {
2117         \g_lngx_ipa_sans_small_caps_tl
2118     }
2119     \tl_if_empty:cF {
2120         g _ lngx _ ipa _ sans _ small _ caps _ features _
2121         tl
2122     } {
2123         SmallCapsFeatures = {
2124             \g_lngx_ipa_sans_small_caps_features_tl
2125         }
2126     }
2127 }
2128 },

```

```

2129 ipa~
2130 mono~ upright~ features = {
2131     UprightFont           = {
2132         \g_lngx_ipa_mono_bold_upright_tl
2133     },
2134     UprightFeatures       = {
2135         \g_lngx_ipa_mono_bold_upright_features_tl
2136     },
2137     BoldFont              = {
2138         \g_lngx_ipa_mono_bold_upright_tl
2139     },
2140     BoldFeatures          = {
2141         \g_lngx_ipa_mono_bold_upright_features_tl
2142     },
2143     ItalicFont            = {
2144         \g_lngx_ipa_mono_italic_tl
2145     },
2146     ItalicFeatures        = {
2147         \g_lngx_ipa_mono_italic_features_tl
2148     },
2149     BoldItalicFont        = {
2150         \g_lngx_ipa_mono_bold_italic_tl
2151     },
2152     BoldItalicFeatures    = {
2153         \g_lngx_ipa_mono_bold_italic_features_tl
2154     },
2155     \tl_if_empty:cF {
2156         g _ lngx _ ipa _ mono _ slanted _ tl
2157     } {
2158         SlantedFont        = {
2159             \g_lngx_ipa_mono_slanted_tl
2160         },
2161         \tl_if_empty:cF {
2162             g _ lngx _ ipa _ mono _ slanted _ features _ tl
2163         } {
2164             SlantedFeatures = {
2165                 \g_lngx_ipa_mono_slanted_features_tl
2166             },
2167         }
2168     }
2169     \tl_if_empty:cF {
2170         g _ lngx _ ipa _ mono _ bold _ slanted _ tl
2171     } {
2172         BoldSlantedFont    = {
2173             \g_lngx_ipa_mono_bold_slanted_tl
2174         },
2175         \tl_if_empty:cF {
2176             g _ lngx _ ipa _ mono _ bold _ slanted _ features
2177             _ tl
2178         } {
2179             BoldSlantedFeatures = {
2180                 \g_lngx_ipa_mono_bold_slanted_features_tl
2181             },
2182         }

```

```

2183     }
2184     \tl_if_empty:cF {
2185         g _ lngx _ ipa _ mono _ swash _ tl
2186     } {
2187         SwashFont = {
2188             \g_lngx_ipa_mono_swash_tl
2189         },
2190         \tl_if_empty:cF {
2191             g _ lngx _ ipa _ mono _ swash _ features _ tl
2192         } {
2193             SwashFeatures = {
2194                 \g_lngx_ipa_mono_swash_features_tl
2195             },
2196         }
2197     }
2198     \tl_if_empty:cF {
2199         g _ lngx _ ipa _ mono _ bold _ swash _ tl
2200     } {
2201         BoldSwashFont = {
2202             \g_lngx_ipa_mono_bold_swash_tl
2203         },
2204         \tl_if_empty:cF {
2205             g _ lngx _ ipa _ mono _ bold _ swash _ features _
2206             tl
2207         } {
2208             BoldSwashFeatures = {
2209                 \g_lngx_ipa_mono_bold_swash_features_tl
2210             },
2211         }
2212     }
2213     \tl_if_empty:cF {
2214         g _ lngx _ ipa _ mono _ small _ caps _ tl
2215     } {
2216         SmallCapsFont = {
2217             \g_lngx_ipa_mono_small_caps_tl
2218         }
2219         \tl_if_empty:cF {
2220             g _ lngx _ ipa _ mono _ small _ caps _ features _
2221             tl
2222         } {
2223             SmallCapsFeatures = {
2224                 \g_lngx_ipa_mono_small_caps_features_tl
2225             }
2226         }
2227     }
2228 }
2229 }
2230 \tl_if_in:cnT {
2231     g _ lngx _ ipa _ upright _ tl
2232 } { NewCM } {
2233     \lngx_set_keys:n {
2234         ipa~
2235         upright~ features = {
2236             IgnoreFontspecFile,

```



```

2237         StylisticSet           = { 5 }
2238     }
2239 }
2240
2241 \tl_if_in:cnT {
2242     g _ lngx _ ipa _ sans _ upright _ tl
2243 } { NewCM } {
2244     \lngx_set_keys:n {
2245         ipa~ sans~
2246         upright~ features           = {
2247             IgnoreFontspecFile,
2248             StylisticSet           = { 5 }
2249         }
2250     }
2251 }
2252 \tl_if_in:cnT {
2253     g _ lngx _ ipa _ mono _ upright _ tl
2254 } { NewCM } {
2255     \lngx_set_keys:n {
2256         ipa~ mono~
2257         upright~ features           = {
2258             IgnoreFontspecFile,
2259             StylisticSet           = { 5 }
2260         }
2261     }
2262 }
2263 \lngx_set_main_ipa_font:ee {
2264     \g_lngx_ipa_upright_features_tl
2265 } {
2266     \g_lngx_ipa_upright_tl
2267 }
2268 \lngx_set_sans_ipa_font:ee {
2269     \g_lngx_ipa_sans_upright_features_tl
2270 } {
2271     \g_lngx_ipa_sans_upright_tl
2272 }
2273 \lngx_set_mono_ipa_font:ee {
2274     \g_lngx_ipa_mono_upright_features_tl
2275 } {
2276     \g_lngx_ipa_mono_upright_tl
2277 }
2278 }
2279 </ipa>

```

```

2280 (*logos)
2281 \ProvidesExplPackage{linguistix-logos}
2282     {2025-05-20}
2283     {v0.1b}
2284     {%
2285     Logos of the ‘Linguistix’ bundle..%
2286     }

```

The fontspec package (if not already loaded).

```

2287
2288 \IfPackageLoadedF { fontspec } {
2289     \RequirePackage { fontspec }
2290 }

```

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

```

2291
2292 \newfontfamily \lngx_logo_font: [
2293     IgnoreFontspecFile,
2294     UprightFont          = { NewCMUncial10-Book.otf },
2295     UprightFeatures      = {
2296         SizeFeatures      = {
2297             {
2298                 Size          = {-8},
2299                 Font          = {NewCMUncial08-Book.otf}
2300             },
2301             {
2302                 Size          = {8-},
2303                 Font          = {NewCMUncial10-Book.otf}
2304             },
2305         }
2306     },
2307     BoldFont             = { NewCMUncial10-Bold.otf },
2308     BoldFeatures         = {
2309         SizeFeatures      = {
2310             {
2311                 Size          = {-8},
2312                 Font          = {NewCMUncial08-Bold.otf}
2313             },
2314             {
2315                 Size          = {8-},
2316                 Font          = {NewCMUncial10-Bold.otf}
2317             },
2318         }
2319     },
2320     Renderer             = { HarfBuzz }
2321 ]{ NewCMUncial10-Book.otf }

```

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

lngx_purple_color

```

2322
2323 \color_set:nn { lngx _ purple _ color } { blue ! 50 ! red }

```

(End of definition for lngx_purple_color. This function is documented on page 14.)

\lngxlogo

```
2324
2325 \NewDocumentCommand \lngxlogo { 0{} } {%
2326   \group_begin:
2327   \lngx_logo_font:
2328   LinguisTi
2329   \color_group_begin:
2330   \color_select:n { lngx_purple_color }
2331   X
2332   \color_group_end:
2333   \IfBlankF { #1 } { - #1 }
2334   \group_end:
2335 }
```

(End of definition for \lngxlogo. This function is documented on page 8.)

\lngxpkg

```
2336
2337 \cs_new:Npn \lngxpkg {
2338   \IfPackageLoadedTF { hyperref } {
2339     \texorpdfstring {
2340       \lngxlogo
2341     } {
2342       LinguisTiX
2343     }
2344   } {
2345     \lngxlogo
2346   }
2347 }
```

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

\lngxbaselogo

\lngxfontslogo

\lngxipalogo

\lngxlogoslogo

\lngxnfsslogo

```
2348
2349 \clist_map_inline:nn {
2350   base,
2351   examples,
2352   fonts,
2353   ipa,
2354   logos,
2355   nfss
2356 } {
2357   \cs_new:cpn { lngx #1 logo } {
2358     \texorpdfstring {
2359       \lngxlogo [ #1 ]
2360     } {
2361       LinguisTiX - #1
2362     }
2363   }
2364 }
2365 </logos>
```

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

```

2366 <nfss>
2367 \ProvidesExplPackage{linguistix-nfss}
2368 {2025-05-20}
2369 {v0.1b}
2370 {%
2371     An extension to the core NFSS commands
2372     from the 'Linguistix' bundle.%
2373 }

```

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.

```

2374
2375 \tl_new:N \l__lngx_normalfont_tmp_tl
2376 \tl_new:N \l__lngx_selectfont_tmp_tl
2377 \tl_new:N \l__lngx_family_tmp_tl
2378 \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.

```

2379
2380 \tl_new:N \l_lngx_current_encoding_tl
2381 \tl_new:N \l_lngx_current_meta_family_tl
2382 \tl_new:N \l_lngx_current_super_family_tl
2383 \tl_new:N \l_lngx_current_series_tl
2384 \tl_new:N \l_lngx_current_shape_tl

```

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

```

\c_lngx_default_rmdefault_tl
\c_lngx_default_sfdefault_tl
\c_lngx_default_ttdefault_tl
2385
2386 \hook_gput_code:nnn { begindocument / end } { . } {
2387     \tl_const:Ne \c_lngx_default_rmdefault_tl { \rmdefault }
2388     \tl_const:Ne \c_lngx_default_sfdefault_tl { \sfdefault }
2389     \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 14.)

```

\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
2390 \tl_set:Nn \l_lngx_current_super_family_tl { default }

```

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

The current encoding is saved in the relevant t_l.

```

2391 \tl_set:Ne \l_lngx_current_encoding_tl {
2392     \encodingdefault
2393 }

```

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.

```

2394 \IfClassLoadedTF { beamer } {
2395     \tl_set:Ne \l_lngx_current_meta_family_tl { sf }
2396 } {
2397     \tl_set:Ne \l_lngx_current_meta_family_tl { rm }
2398 }

```

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

```
2399 \tl_set:Nc \l_lngx_current_series_tl { md }
2400 \tl_set:Nc \l_lngx_current_shape_tl { up }
2401 }
```

(End of definition for `\l_lngx_current_encoding_tl` and others. These functions are documented on page 14.)

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

```
2402
2403 \hook_gput_code:nnn { cmd / normalfont / before } { . } {
2404   \tl_set:Nc \l__lngx_normalfont_tmp_tl { \f@encoding }
2405 }
```

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`.

```
2406
2407 \hook_gput_code:nnn { cmd / normalfont / after } { . } {
2408   \tl_set_eq:NN \l_lngx_current_encoding_tl
2409     \l__lngx_normalfont_tmp_tl
2410   \tl_clear:N \l__lngx_normalfont_tmp_tl
2411 }
```

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

```
2412
2413 \hook_gput_code:nnn { cmd / selectfont / before } { . } {
2414   \tl_set:Nc \l__lngx_selectfont_tmp_tl { \f@encoding }
2415 }
2416
2417 \hook_gput_code:nnn { cmd / selectfont / after } { . } {
2418   \tl_set_eq:NN \l_lngx_current_encoding_tl
2419     \l__lngx_selectfont_tmp_tl
2420   \tl_clear:N \l__lngx_selectfont_tmp_tl
2421 }
```

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.

```
2422
2423 \hook_gput_code:nnn { cmd / rmfamily / before } { . } {
2424   \tl_set:Nc \l_lngx_current_meta_family_tl { rm }
2425   \tl_set:Nc \l__lngx_family_tmp_tl { \f@encoding }
2426 }
2427
2428 \hook_gput_code:nnn { cmd / rmfamily / after } { . } {
2429   \tl_set:Nc \l_lngx_current_meta_family_tl { rm }
2430   \tl_set_eq:NN \l_lngx_current_encoding_tl
2431     \l__lngx_family_tmp_tl
2432   \tl_clear:N \l__lngx_family_tmp_tl
2433 }
2434
2435 \hook_gput_code:nnn { cmd / sffamily / before } { . } {
2436   \tl_set:Nc \l_lngx_current_meta_family_tl { sf }
```

```

2437 \tl_set:Nn \l__lngx_family_tmp_tl { \f@encoding }
2438 }
2439
2440 \hook_gput_code:nnn { cmd / sffamily / after } { . } {
2441   \tl_set:Nn \l_lngx_current_meta_family_tl { sf }
2442   \tl_set_eq:NN \l_lngx_current_encoding_tl
2443     \l__lngx_family_tmp_tl
2444   \tl_clear:N \l__lngx_family_tmp_tl
2445 }
2446
2447 \hook_gput_code:nnn { cmd / ttfamily / before } { . } {
2448   \tl_set:Nn \l_lngx_current_meta_family_tl { tt }
2449   \tl_set:Nn \l__lngx_family_tmp_tl { \f@encoding }
2450 }
2451
2452 \hook_gput_code:nnn { cmd / ttfamily / after } { . } {
2453   \tl_set:Nn \l_lngx_current_meta_family_tl { tt }
2454   \tl_set_eq:NN \l_lngx_current_encoding_tl
2455     \l__lngx_family_tmp_tl
2456   \tl_clear:N \l__lngx_family_tmp_tl
2457 }

```

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

```

2458
2459 \hook_gput_code:nnn { cmd / mdseries / after } { . } {
2460   \tl_set:Nn \l_lngx_current_series_tl { md }
2461 }
2462
2463 \hook_gput_code:nnn { cmd / bfseries / after } { . } {
2464   \tl_set:Nn \l_lngx_current_series_tl { bf }
2465 }

```

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

```

2466
2467 \hook_gput_code:nnn { cmd / upshape / after } { . } {
2468   \tl_set:Nn \l_lngx_current_shape_tl { up }
2469 }
2470
2471 \hook_gput_code:nnn { cmd / itshape / after } { . } {
2472   \tl_set:Nn \l_lngx_current_shape_tl { it }
2473 }
2474
2475 \hook_gput_code:nnn { cmd / scshape / after } { . } {
2476   \tl_set:Nn \l_lngx_current_shape_tl { sc }
2477 }
2478
2479 \hook_gput_code:nnn { cmd / sscshape / after } { . } {
2480   \tl_set:Nn \l_lngx_current_shape_tl { ssc }
2481 }
2482
2483 \hook_gput_code:nnn { cmd / slshape / after } { . } {
2484   \tl_set:Nn \l_lngx_current_shape_tl { sl }

```

```

2485 }
2486
2487 \hook_gput_code:nnn { cmd / swshape / after } { . } {
2488   \tl_set:Nn \l_lngx_current_shape_tl { sw }
2489 }
2490
2491 \hook_gput_code:nnn { cmd / ulcshape / after } { . } {
2492   \tl_set:Nn \l_lngx_current_shape_tl { ulc }
2493 }
2494
2495 \hook_gput_code:nnn { cmd / ulcshape / after } { . } {
2496   \tl_set:Nn \l_lngx_current_shape_tl { #1 }
2497 }

```

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

```

2498
2499 \prg_new_conditional:Nnn \lngx_if_encoding:n {
2500   p,
2501   T,
2502   F,
2503   TF
2504 } {
2505   \tl_if_eq:NnTF \l_lngx_current_encoding_tl { #1 } {
2506     \prg_return_true:
2507   } {
2508     \prg_return_false:
2509   }
2510 }
2511

```

(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`

```

2512
2513 \cs_new_eq:NN \IfEncodingTF \lngx_if_encoding:nTF
2514 \cs_new_eq:NN \IfEncodingT \lngx_if_encoding:nT
2515 \cs_new_eq:NN \IfEncodingF \lngx_if_encoding:nF

```

(End of definition for `\IfEncodingTF`, `\IfEncodingT`, and `\IfEncodingF`. These functions are documented on page 10.)

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

`\lngx_if_meta_family:nTF`

```

2516
2517 \prg_new_conditional:Nnn \lngx_if_meta_family:n {
2518   p,
2519   T,
2520   F,
2521   TF
2522 } {
2523   \tl_if_eq:NnTF \l_lngx_current_meta_family_tl { #1 } {
2524     \prg_return_true:
2525   } {
2526     \prg_return_false:
2527   }
2528 }

```

(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`

2529

2530 `\cs_new_eq:NN \IfMetaFamilyTF \lngx_if_meta_family:nTF`

2531 `\cs_new_eq:NN \IfMetaFamilyT \lngx_if_meta_family:nT`

2532 `\cs_new_eq:NN \IfMetaFamilyF \lngx_if_meta_family:nF`

(End of definition for `\IfMetaFamilyTF`, `\IfMetaFamilyT`, and `\IfMetaFamilyF`. These functions are documented on page 10.)

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

`\lngx_if_super_family:nTF`

2533

2534 `\prg_new_conditional:Nnn \lngx_if_super_family:n {`

2535 `P,`

2536 `T,`

2537 `F,`

2538 `TF`

2539 `} {`

2540 `\tl_if_eq:NnTF \l_lngx_current_super_family_tl { #1 } {`

2541 `\prg_return_true:`

2542 `} {`

2543 `\prg_return_false:`

2544 `}`

2545 `}`

(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`

2546

2547 `\cs_new_eq:NN \IfSuperFamilyTF \lngx_if_super_family:nTF`

2548 `\cs_new_eq:NN \IfSuperFamilyT \lngx_if_super_family:nT`

2549 `\cs_new_eq:NN \IfSuperFamilyF \lngx_if_super_family:nF`

(End of definition for `\IfSuperFamilyTF`, `\IfSuperFamilyT`, and `\IfSuperFamilyF`. These functions are documented on page 10.)

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

`\lngx_if_series:nTF`

2550

2551 `\prg_new_conditional:Nnn \lngx_if_series:n {`

2552 `P,`

2553 `T,`

2554 `F,`

2555 `TF`

2556 `} {`

2557 `\tl_if_eq:NnTF \l_lngx_current_series_tl { #1 } {`

2558 `\prg_return_true:`

2559 `} {`

2560 `\prg_return_false:`

2561 `}`

2562 `}`

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

`\IfSeriesTF` Its user-side macros.

`\IfSeriesT`

`\IfSeriesF`

```

2563
2564 \cs_new_eq:NN \IfSeriesTF \lngx_if_series:nTF
2565 \cs_new_eq:NN \IfSeriesT \lngx_if_series:nT
2566 \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`

```

2567
2568 \prg_new_conditional:Nnn \lngx_if_shape:n {
2569   p,
2570   T,
2571   F,
2572   TF
2573 } {
2574   \tl_if_eq:NnTF \l_lngx_current_shape_tl { #1 } {
2575     \prg_return_true:
2576   } {
2577     \prg_return_false:
2578   }
2579 }

```

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

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

`\IfShapeT`

`\IfShapeF`

```

2580
2581 \cs_new_eq:NN \IfShapeTF \lngx_if_shape:nTF
2582 \cs_new_eq:NN \IfShapeT \lngx_if_shape:nT
2583 \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.

```

2584
2585 \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:`

`\lngx_if_meta_family_sf:TF`

`\lngx_if_meta_family_tt_p:`

`\lngx_if_meta_family_tt:TF`

```

2586
2587 \clist_map_inline:nn {
2588   rm,
2589   sf,
2590   tt
2591 } {
2592   \prg_new_conditional:cnn {
2593     lngx _ if _ meta _ family _ #1 :
2594   } {
2595     p, T, F, TF
2596   } {
2597     \tl_if_eq:NnTF \l_lngx_current_meta_family_tl { #1 } {
2598       \prg_return_true:

```

```

2599     } {
2600         \prg_return_false:
2601     }
2602 }
2603 }

```

(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.)

`\lngx_if_series_md_p:` Separate conditionals for both the series.

```

\lngx_if_series_md:TF 2604
\lngx_if_series_md:TF 2605 \clist_map_inline:nn {
\lngx_if_series_bf_p: 2606     md,
\lngx_if_series_bf:TF 2607     bf
2608 } {
2609     \prg_new_conditional:cnn { lngx _ if _ series _ #1 : } {
2610         p, T, F, TF
2611     } {
2612         \tl_if_eq:NnTF \l_lngx_current_series_tl { #1 } {
2613             \prg_return_true:
2614         } {
2615             \prg_return_false:
2616         }
2617     }
2618 }

```

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

`\lngx_if_shape_up_p:` Separate conditionals for all the shapes.

```

\lngx_if_shape_up:TF 2619
\lngx_if_shape_it_p: 2620 \clist_map_inline:nn {
\lngx_if_shape_it:TF 2621     up,
\lngx_if_shape_sc_p: 2622     it,
\lngx_if_shape_sc:TF 2623     sc,
\lngx_if_shape_ssc_p: 2624     ssc,
\lngx_if_shape_ssc:TF 2625     sl,
\lngx_if_shape_sl_p: 2626     sw,
\lngx_if_shape_sl:TF 2627     ulc
2628 } {
2629     \prg_new_conditional:cnn { lngx _ if _ shape _ #1 : } {
2630         p, T, F, TF
2631     } {
2632         \tl_if_eq:NnTF \l_lngx_current_shape_tl { #1 } {
2633             \prg_return_true:
2634         } {
2635             \prg_return_false:
2636         }
2637     }
2638 }

```

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

These keys are used in the argument of `\lngx_super_font_family:nn`. This is why they are separated from the set `lngx_keys`. We create new tls using these keys that

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.

```

2639
2640 \clist_map_inline:nn {
2641     rm,
2642     sf,
2643     tt
2644 } {
2645     \keys_define:nn { lngx _ nfss } {
2646         #1
2647         .code:n          = {
2648             \tl_gclear_new:c {
2649                 g _ lngx _ \l__lngx_nfss_tmp_tl _ #1 default _ tl
2650             }
2651             \tl_gset:cn {
2652                 g _ lngx _ \l__lngx_nfss_tmp_tl _ #1 default _ tl
2653             } { ##1 }
2654         }
2655     }
2656 }

```

`\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`

```

2657
2658 \cs_new_protected:Npn \lngx_super_font_family:nn #1#2 {
2659     \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.

```

2660     \keys_set:nn { lngx _ nfss } { #2 }
2661     \tl_clear:N \l__lngx_nfss_tmp_tl
2662 }
2663
2664 \cs_set_eq:NN \superfontfamily
2665     \lngx_super_font_family:nn

```

(End of definition for `\lngx_super_font_family:nn` and `\superfontfamily`. These functions are documented on page 15.)

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

```

2666
2667 \cs_new_protected:Npn \lngx_soft_super_font_family:nn #1#2 {
2668     \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.

```

2669     \clist_map_inline:nn {
2670         rm,
2671         sf,
2672         tt
2673     } {
2674         \tl_if_empty:cF { g _ lngx _ #1 _ ##1 default _ tl } {
2675             \cs_set:cpe { ##1 default } {

```

```

2676         \tl_use:c { g _ lngx _ #1 _ ##1 default _ tl }
2677     }
2678 }
2679 }

```

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

```

2680 \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.

```

2681 \clist_map_inline:nn { #2 } {
2682     \str_case:nn { ##1 } {
2683         { encoding } {
2684             \exp_args:NV \fontencoding
2685                 \l_lngx_current_encoding_tl
2686         }
2687         { family } {
2688             \use:c {
2689                 \l_lngx_current_meta_family_tl family
2690             }
2691             \exp_args:NV \fontencoding
2692                 \l_lngx_current_encoding_tl
2693             \selectfont
2694         }
2695         { series } {
2696             \use:c {
2697                 \l_lngx_current_series_tl series
2698             }
2699         }
2700         { shape } {
2701             \use:c {
2702                 \l_lngx_current_shape_tl shape
2703             }
2704         }
2705     }
2706 }
2707 }
2708
2709 \cs_set_eq:NN \softsuperfontfamily
2710         \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.

```

2711
2712 \cs_new_protected:Npn \lngx softer super font family:n #1 {
2713     \lngx_soft_super_font_family:nn { #1 } {
2714         family,
2715         series,
2716         shape

```

```

2717 }
2718 }
2719
2720 \cs_set_eq:NN \softersuperfontfamily
2721 \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` This function resets all the attributes. It is available as a user-side macro.
`\softestsuperfontfamily`

```

2722
2723 \cs_new_protected:Npn \lngx_softest_super_font_family:n #1 {
2724   \lngx_soft_super_font_family:nn { #1 } {
2725     encoding,
2726     family,
2727     series,
2728     shape
2729   }
2730 }
2731
2732 \cs_set_eq:NN \softestsuperfontfamily
2733 \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` Following the same logic, I now provide the command for resetting to the default super
`\softnormalfont` family, but retaining the active attributes. I provide a user-side macro for this.

```

2734
2735 \cs_new_protected:Npn \lngx_soft_normal_font:n #1 {
2736   \tl_set:Nc \l_lngx_current_super_family_tl { default }
2737   \clist_map_inline:nn {
2738     rm,
2739     sf,
2740     tt
2741   } {
2742     \cs_set:cpe { ##1 default } {
2743       \tl_use:c { c _ lngx _ default _ ##1 default _ tl }
2744     }
2745   }
2746   \normalfont
2747   \clist_map_inline:nn { #1 } {
2748     \str_case:nn { ##1 } {
2749       { encoding } {
2750         \exp_args:NV \fontencoding
2751           \l_lngx_current_encoding_tl
2752       }
2753       { family } {
2754         \use:c {
2755           \l_lngx_current_meta_family_tl family
2756         }
2757         \exp_args:NV \fontencoding
2758           \l_lngx_current_encoding_tl
2759       }
2760       \selectfont
2761     }

```

```

2761     { series } {
2762       \use:c {
2763         \l_lngx_current_series_tl series
2764       }
2765     }
2766     { shape } {
2767       \use:c {
2768         \l_lngx_current_shape_tl shape
2769       }
2770     }
2771   }
2772 }
2773 }
2774
2775 \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`

```

2776
2777 \cs_new_protected:Npn \lngx_softest_normal_font: {
2778   \lngx_soft_normal_font:n {
2779     family,
2780     series,
2781     shape
2782   }
2783 }
2784
2785 \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`

```

2786
2787 \cs_new_protected:Npn \lngx_softest_normal_font: {
2788   \lngx_soft_normal_font:n {
2789     encoding,
2790     family,
2791     series,
2792     shape
2793   }
2794 }
2795
2796 \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.

```

2797 \cs_new:Npn \CurrentEncoding {
2798   \tl_use:N \l_lngx_current_encoding_tl
2799 }

```

```

2800 \cs_new:Npn \CurrentMetaFamily {
2801   \tl_use:N \l_lngx_current_meta_family_tl
2802 }
2803 \cs_new:Npn \CurrentSuperFamily {
2804   \tl_use:N \l_lngx_current_super_family_tl
2805 }
2806 \cs_new:Npn \CurrentSeries {
2807   \tl_use:N \l_lngx_current_series_tl
2808 }
2809 \cs_new:Npn \CurrentShape {
2810   \tl_use:N \l_lngx_current_shape_tl
2811 }
2812 </nfss>

```

(End of definition for `\CurrentEncoding` and others. These functions are documented on page [10](#).)

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.