

The LINGUISṬIX bundle

निरंजन

22 December 2025 (vo.6a)

🏠 <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	8	LINGUISṬIX-ipa	7
2	Planned	4		Interface... 15; Implementation... 51	
3	Funding	4	9	LINGUISṬIX-LANGUAGES	9
4	Acknowledgements	4		Interface... 16; Implementation... 77	
5	LINGUISṬIX-BASE	5	10	LINGUISṬIX-LOGOS	11
	Interface... 15; Implementation... 20			Interface... 16; Implementation... 84	
6	LINGUISṬIX-FIXPEX	5	11	LINGUISṬIX-NFSS	12
	Interface... 15; Implementation... 21			Interface... 17; Implementation... 86	
7	LINGUISṬIX-FONTS	5		GNU Free Documentation License	98
	Interface... 15; Implementation... 23				

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 Phonetic symbols

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

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

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

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

2 Planned

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

3 Funding

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

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

4 Acknowledgements

This package relies the most on the New Computer Modern font family. I would like to express my gratitude to Antonis Tsolomitis who tirelessly worked on the set of IPA symbols and brought back the good old charm of TIPA's design in the modern Unicode world. I would like to thank Renuka and Avinash who taught me linguistics. They nourished my passion, helped me pursue my love for the subject as well as the computation that came along with it. I could have never imagined myself working on a package written in \LaTeX_3 's syntax. Not so long ago, I used to find it very complicated. It's mostly Jonathan Spratte and Florent Rougon's help (and, at times, scolding :P) that helped me get used to it. I would also like to mention C.V. Radhakrishnan for being an important part of my journey in \LaTeX . Lastly, to all the free software people who have created this friendly and supportive world for people by investing their precious time and resources!

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

Documentation

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

```
\usepackage{linguistix}
```

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

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

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

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

 $\text{\textcolor{violet}{l}}\text{\textcolor{violet}{i}}\text{\textcolor{violet}{n}}\text{\textcolor{violet}{g}}\text{\textcolor{violet}{u}}\text{\textcolor{violet}{i}}\text{\textcolor{violet}{s}}\text{\textcolor{violet}{t}}\text{\textcolor{violet}{i}}\text{\textcolor{violet}{x}}$ $\{ \langle \textit{key-value-list} \rangle \}$

We have a single set of keys for the entire bundle. Each package appends keys to the same set. The argument of this central processor command is the comma-separated $\langle \textit{key-value-list} \rangle$. So you can load any package of LINGUIS $\text{\textcolor{violet}{T}}$ $\text{\textcolor{violet}{I}}$ $\text{\textcolor{violet}{X}}$ and use the $\text{\textcolor{violet}{l}}\text{\textcolor{violet}{i}}\text{\textcolor{violet}{n}}\text{\textcolor{violet}{g}}\text{\textcolor{violet}{u}}\text{\textcolor{violet}{i}}\text{\textcolor{violet}{s}}\text{\textcolor{violet}{t}}\text{\textcolor{violet}{i}}\text{\textcolor{violet}{x}}$ command. The only exception to this is LINGUIS $\text{\textcolor{violet}{T}}$ $\text{\textcolor{violet}{I}}$ $\text{\textcolor{violet}{X}}$ -NFSS. We will see how it is different in its section.

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

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

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

 $\text{\textcolor{violet}{u}}\text{\textcolor{violet}{m}}\text{\textcolor{violet}{g}}\text{\textcolor{violet}{l}}\text{\textcolor{violet}{a}}$ This is a replica of the `unicode-math-\text{\textcolor{violet}{g}}\text{\textcolor{violet}{l}}\text{\textcolor{violet}{a}}`. Since the `expex-\text{\textcolor{violet}{g}}\text{\textcolor{violet}{l}}\text{\textcolor{violet}{a}}` is more relevant in linguistics, I set it as the default. If one needs to use `unicode-math-\text{\textcolor{violet}{g}}\text{\textcolor{violet}{l}}\text{\textcolor{violet}{a}}`, they can use this command.

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

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

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

Antonis suggested a typographic enhancement for the logo of $\text{\textcolor{violet}{L}}\text{\textcolor{violet}{A}}\text{\textcolor{violet}{T}}\text{\textcolor{violet}{E}}\text{\textcolor{violet}{X}}$. The default logo scales the ‘A’ and that affects the ‘colour’ of the font. This is why I renew the logo with the code given by Antonis. The original logo is also available with an alternative command.

 $\text{\textcolor{violet}{L}}\text{\textcolor{violet}{a}}\text{\textcolor{violet}{T}}\text{\textcolor{violet}{e}}\text{\textcolor{violet}{X}}$ $\text{\textcolor{violet}{L}}\text{\textcolor{violet}{A}}\text{\textcolor{violet}{T}}\text{\textcolor{violet}{E}}\text{\textcolor{violet}{X}}$
 $\text{\textcolor{violet}{l}}\text{\textcolor{violet}{o}}\text{\textcolor{violet}{g}}\text{\textcolor{violet}{L}}\text{\textcolor{violet}{a}}\text{\textcolor{violet}{T}}\text{\textcolor{violet}{e}}\text{\textcolor{violet}{X}}$ $\text{\textcolor{violet}{L}}\text{\textcolor{violet}{A}}\text{\textcolor{violet}{T}}\text{\textcolor{violet}{E}}\text{\textcolor{violet}{X}}$

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

1 Text

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

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

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

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

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

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

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

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

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

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

2 Maths

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

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

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

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

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

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

8 LINGUIS τ I χ -IPA

L^AT_EX₃-interface | Implementation

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

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

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

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 `[aɪ phi: eɪ]` and `/aɪ phi: eɪ/` 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`

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

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

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

Continued on the next page...

Family	LINGUISŢIX-FONTS	LINGUISŢIX-IPA
	text mono slanted features	ipa mono slanted features
	text mono bold slanted	ipa mono bold slanted
	text mono bold slanted features	ipa mono bold slanted features
	text mono swash	ipa mono swash
	text mono swash features	ipa mono swash features
	text mono bold swash	ipa mono bold swash
	text mono bold swash features	ipa mono bold swash features
	text mono small caps	ipa mono small caps
	text mono small caps features	ipa mono small caps features
<i>End of the table...</i>		

Table 1: Font keys provided by LINGUISŢIX-FONTS and LINGUISŢIX-IPA

Apart from these, both the packages provide the following keys for appending to the extra features for the respective fonts:

- text extra features
- text sans extra features
- text mono extra features
- ipa extra features
- ipa sans extra features
- ipa mono extra features

9 LINGUISŢIX-LANGUAGES

L^AT_EX₃-interface | Implementation

This package is intended to provide support for loading Unicode fonts as well as other necessary settings for using languages. It is a wrapper around the `babel` package, but it provides some other useful settings which `babel` doesn't agree to add. This package is a little opinionated and pushes for 'modern' practices e.g., Unicode, Lua^LA^TE^X, no-markup multilingual text etc. As of now, only a little support is available. If you want your language to be supported, you can ask for support at the bug tracker of the repository or you can send an email in the public mailing list for the project. You may subscribe to the mailing list at: %mail.gnu.org.ua/mailman/listinfo/linguistix-languages%. Here, I list down some L^AT_EX-aspects that may demand some modifications in the default settings.

Fonts: The package works with Unicode and does not worry about legacy methods. If you want support for your language, first and foremost, you should let me know standard OpenType fonts suitable for your language. Note that they should be freely licensed. I won't support proprietary software with LINGUISŢIX.

babel support: As mentioned before, the package adds on to the support provided by package `babel`. So check if the language files – specifically the modern `.ini` files – have the correct settings. Sometimes they may need to undergo native-speaker's scrutiny. Whatever is wrong in `babel`, may not get corrected in LINGUISŢIX.

Numbers: L^AT_EX uses a lot of counters and all of them, by default, print Latin numerals/characters. E.g., `\arabic{page}` prints the page number in Latin, but `\roman{page}` prints the same in Roman convention, i.e., ‘i, ii, ...’. Does your language allow them? E.g., Greek doesn’t like Latin alphabets, but doesn’t mind Roman numerals. Instead of Latin alphabets, Greek prefers to use its own numeral system. Marathi doesn’t like any of these, but it doesn’t have alternative forms of numeration, so it changes certain cases drastically. E.g., By default, in nested `enumerate` environment, L^AT_EX provides Roman numerals and Latin alphabets. Since Marathi doesn’t have good alternatives, it renews the printing of nested `\items` as `I`, `I.I`, `I.I.I` and `I.I.I.I`. This is reset to defaults when the language is changed. Keeping this in mind, I am listing down some places where I found non-native numbering (I might have missed something in which case it deserves to be reported as a bug, so feel free to do so!).

1. Page numbers (in front matter, main matter).
2. Part numbers.
3. Second, third and fourth levels of enumeration.

ExPex: Labels provided by ExPex package (see: tex.stackexchange.com/a/548668).

Typography: Language-specific conventions like using Italic for emphasis. It is a Latin-script specific convention (note that I don’t mean slanted when I say Italic). Different languages have different conventions of emphasising (e.g., Marathi uses bold font for emphasis).

Miscellaneous: Anything other than these.

I am very much willing to support multilingual typesetting for multiple languages, but I need to know the things mentioned in this list in order to provide the best suited output. Please consider submitting a detailed feature request. The documentation of supported languages is in separate PDFs. This documentation only describes the user-side commands provided by the package.

<code>languages</code>	<code>{\list of languages}</code>
<code>\loadlanguages</code>	<code>{\list of languages}</code>

This key works with the central key-parser of L^INGUISTI_X, i.e., `\linguistix`. It accepts one argument that is a list of languages user wants to load. Unlike `babel`, the first element of this list is set as the main language for the document. The command `\loadlanguages` has the identical behaviour. In fact, it is a wrapper around the key.

<code>\providelanguage</code>	<code>{\language options}</code>	<code>{\language name}</code>
-------------------------------	----------------------------------	-------------------------------

This is a wrapper command over `\babelprovide`. The first argument is passed to the optional argument of `\babelprovide` and the second one to the mandatory argument of the same. For more information, please read `babel`’s manual.

Languages supported by L^INGUISTI_X-LANGUAGES are loaded with an `.ldf` file. If it is absent, the package produces a warning.

<code>native numbering</code>	<code>= {\strict/logical/off}</code>
-------------------------------	--------------------------------------

Many languages need native digits. Adding them in a multilingual document is quite complicated. This key sets the plugs provided for the socket of the same name. Language packages already take care of them, but if you want to change anything mid-document, you can use this key. It has three choices available as its value as seen below.

strict The ‘strict’ plug changes the `\lngx_counter:n` command to the counter of the main language of the document. That way all the counters are printed in the main language.

logical This plug changes the meaning of `\lngx_counter:n` to the `\localecounter` command provided by `babel`. It picks up the surrounding language and uses its native digits. E.g., when Marathi is being typeset, it will print counters in Marathi. When it is changed to English, it will start printing the same in English. Note that this will reflect in table of contents/tables/figures too. It is called logical numbering because it obeys \TeX ’s logic more than what is generally considered the standard. E.g., imagine you have an English section followed by a Marathi section on the same page. Both of them will follow their own numerals for default \TeX counters. Since both of them are on the same page, while shipping out, the last active language will be used for processing the page number (Marathi in this case). This creates a table of contents with Latin numeral as the section counter, but Marathi numeral as the page number. Only experiments can determine if an option like this can have valid use-cases, so it is provided. If you use it, be aware that the results might not be the most pleasant to your aesthetic values. They are so because of the logic of \TeX .

off It is equivalent of the `noop` plug when the other two are not used at all. It is only required when you want to go back to \LaTeX defaults. E.g., if you have turned strict native numbering in some language and you want it to go back to \LaTeX defaults, you may use this.

IO LINGUIS $\text{\textcolor{violet}{T}}$ $\text{\textcolor{violet}{I}}$ $\text{\textcolor{violet}{X}}$ -LOGOS

\LaTeX 3-interface | Implementation

This is a small package that provides commands for printing logos of the LINGUIS $\text{\textcolor{violet}{T}}$ $\text{\textcolor{violet}{I}}$ $\text{\textcolor{violet}{X}}$ bundle. The logo is printed in New Computer Modern Uncial font. It uses purple colour for the ‘X’ in it and it is defined using `l3color` module. It provides one command that takes an optional argument. Obviously it is ‘protected’. It is as follows:

\lngxlogo [*(package name)*]

The logo of the *(package name)* from the LINGUIS $\text{\textcolor{violet}{T}}$ $\text{\textcolor{violet}{I}}$ $\text{\textcolor{violet}{X}}$ bundle is printed with this command, e.g., `\lngxlogo{fonts}` \longrightarrow LINGUIS $\text{\textcolor{violet}{T}}$ $\text{\textcolor{violet}{I}}$ $\text{\textcolor{violet}{X}}$ fonts.

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

<code>\lngxpkg</code>	★ LINGUIS $\text{\textcolor{violet}{T}}$ $\text{\textcolor{violet}{I}}$ $\text{\textcolor{violet}{X}}$
<code>\lngxbaselogo</code>	★ LINGUIS $\text{\textcolor{violet}{T}}$ $\text{\textcolor{violet}{I}}$ $\text{\textcolor{violet}{X}}$ -BASE
<code>\lngxfontslogo</code>	★ LINGUIS $\text{\textcolor{violet}{T}}$ $\text{\textcolor{violet}{I}}$ $\text{\textcolor{violet}{X}}$ -FONTS
<code>\lngxipalogo</code>	★ LINGUIS $\text{\textcolor{violet}{T}}$ $\text{\textcolor{violet}{I}}$ $\text{\textcolor{violet}{X}}$ -IPA
<code>\lngxlogoslogo</code>	★ LINGUIS $\text{\textcolor{violet}{T}}$ $\text{\textcolor{violet}{I}}$ $\text{\textcolor{violet}{X}}$ -LOGOS
<code>\lngxnfsslogo</code>	★ LINGUIS $\text{\textcolor{violet}{T}}$ $\text{\textcolor{violet}{I}}$ $\text{\textcolor{violet}{X}}$ -NFSS

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

<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>*</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>\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>*</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>\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>*</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>\IfSeriesTF</code>	<code>* {\langle series \rangle} {\langle true code \rangle} {\langle false code \rangle}</code>
<code>\IfSeriesT</code>	<code>* {\langle series \rangle} {\langle true code \rangle}</code>
<code>\IfSeriesF</code>	<code>* {\langle series \rangle} {\langle false code \rangle}</code>
<code>\CurrentSeries</code>	<code>*</code> If the current series matches with the given $\langle series \rangle$, it selects the true branch and false otherwise. The <code>\CurrentSeries</code> macro expands to the current series.

<code>\IfShapeTF</code>	<code>* {\langle shape \rangle} {\langle true code \rangle} {\langle false code \rangle}</code>
<code>\IfShapeT</code>	<code>* {\langle shape \rangle} {\langle true code \rangle}</code>
<code>\IfShapeF</code>	<code>* {\langle shape \rangle} {\langle false code \rangle}</code>
<code>\CurrentShape</code>	<code>*</code> If the current series matches with the given $\langle shape \rangle$, it selects the true branch and false otherwise. The <code>\CurrentShape</code> macro expands to the current shape.

<code>\superfontfamily</code>	<code>{\family ID} {\rm={\rm NFSS}},sf={\sf NFSS},tt={\tt NFSS}}}</code>
-------------------------------	--------------------------------------------------------------------------

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

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

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

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

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

```

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

\begin{document}
test \lngxipa test \softernormalfont test\par
\makeatletter
\sffamily\itshape\bfseries
\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 \mathcal{T} X-BASE

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

`\lngx_set_keys:n` $\langle keyval list \rangle$

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

LINGUIS \mathcal{T} X-FIXPEX

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

No L^AT_EX₃ function provided by this package.

LINGUIS \mathcal{T} X-FONTS

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

`\g_lngx_old_style_bool`
`\g_lngx_old_style_one_bool`
`\g_lngx_bourbaki_bool`

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.

`\lngx_set_main_font:nn` $\{\langle features \rangle\}$ $\{\langle font \rangle\}$
`\lngx_set_main_font:ee` $\{\langle features \rangle\}$ $\{\langle font \rangle\}$
`\lngx_set_sans_font:nn` $\{\langle features \rangle\}$ $\{\langle font \rangle\}$
`\lngx_set_sans_font:ee` $\{\langle features \rangle\}$ $\{\langle font \rangle\}$

`\lngx_set_mono_font:nn` 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.

`\lngx_set_mono_font:ee`

`\lngx_set_math_font:nn`

`\lngx_set_math_font:ee`

`\lngx_other_main_font:nnn` $\{\langle language \rangle\}$ $\{\langle features \rangle\}$ $\{\langle font \rangle\}$
`\lngx_other_main_font:nee` $\{\langle language \rangle\}$ $\{\langle features \rangle\}$ $\{\langle font \rangle\}$
`\lngx_other_sans_font:nnn` $\{\langle language \rangle\}$ $\{\langle features \rangle\}$ $\{\langle font \rangle\}$
`\lngx_other_sans_font:nee` $\{\langle language \rangle\}$ $\{\langle features \rangle\}$ $\{\langle font \rangle\}$
`\lngx_other_mono_font:nnn`
`\lngx_other_mono_font:nee`

These commands take three arguments. These are wrapper commands around the font-setting commands of `babel`. The $\langle features \rangle$ are passed to the optional argument and the $\langle font \rangle$ is passed to the mandatory argument of the respective command from the aforementioned list.

LINGUIS \mathcal{T} X-IPA

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

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

`\lngx_set_main_ipa_font:nn` $\{\langle features \rangle\}$ $\{\langle font \rangle\}$
`\lngx_set_main_ipa_font:ee`
`\lngx_main_ipa:`
`lngx_ipa_rm_nfss`

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`.

`\lngx_set_sans_ipa_font:nn` $\{\langle features \rangle\}$ $\{\langle font \rangle\}$
`\lngx_set_sans_ipa_font:ee`
`\lngx_sans_ipa:`
`lngx_ipa_sf_nfss`

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

<code>\lngx_set_mono_ipa_font:nn</code>	<code>{\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 accessed with the NFSS family <code>lngx_ipa_nfss_nfss</code> .
<code>lngx_ipa_tt_nfss</code>	

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

LINGUIS \mathcal{T} \mathbf{X} -LANGUAGES

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

Here are the L₃ functions defined for LINGUIS \mathcal{T} \mathbf{X} -LANGUAGES.

<code>\g_lngx_main_language_tl</code>	A <code>tl</code> that globally stores the main language of the document.
---------------------------------------	---------------------------------------------------------------------------

<code>\g_lngx_languages_clist</code>	A <code>clist</code> that globally stores the languages that are used.
--------------------------------------	------------------------------------------------------------------------

<code>\lngx_languages:nn</code>	<code>{\language options}\{language name}</code>
<code>\lngx_languages:VV</code>	<code>\language options tl \language tl</code>
	These functions read the V-type argument provided to them and pass it to the <code>\babelprovide</code> command for loading languages.

<code>\lngx_load_languages:n</code>	<code>{\list of languages}</code>
	This function loads the languages in LINGUIS \mathcal{T} \mathbf{X} sense.

<code>\lngx_counter:n</code>	This is a developers function provided for printing the counter based on the plug selected. It changes the meaning according to the active value of <code>native-numbering</code> socket.
------------------------------	-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------



<code>\lngx_misc_reset:</code>	This function resets a lot of custom settings done by some languages. It has to be used inside <code>\addto</code> macro provided by the <code>babel</code> package.
--------------------------------	----------------------------------------------------------------------------------------------------------------------------------------------------------------------

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

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

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

<code>\lngx_logo_font:</code>	This function switches to the New Computer Modern Uncial font family.
-------------------------------	-----------------------------------------------------------------------

<code>lngx_purple_color</code>	I don't like the default purple colour of the <code>xcolor</code> package (i.e., ). Thus I have created a new colour using <code>l3color</code> module. It can be accessed using this variable. The color looks like:  .
--------------------------------	---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

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

```
\c_lngx_default_rmdefault_tl * These tls 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 tls 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:nn {\family ID} {\rm=\langle rm NFSS \rangle, sf=\langle sf NFSS \rangle, tt=\langle tt NFSS \rangle}

```

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

```

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

```

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

```

\lngx_soft_normal_font:n {\langle ID \rangle}
\lngx_softer_normal_font:
\lngx_softest_normal_font:

```

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

Implementation

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

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

Provide the package with its basic information.

```
1 <*package>
2 \ProvidesExplPackage{linguistix}
3     {2025-12-22}
4     {v0.6a}
5     {%
6         The ‘LinguisTiX’ bundle: Enhanced
7         support for linguistics.%
8     }
```

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

```
9
10 \IfPackageLoadedF { linguistix-base } {
11     \RequirePackage { linguistix-base }
12 }
13 \IfPackageLoadedF { linguistix-fonts } {
14     \RequirePackage { linguistix-fonts }
15 }
16 \IfPackageLoadedF { linguistix-ipa } {
17     \RequirePackage { linguistix-ipa }
18 }
19 \IfPackageLoadedF { linguistix-languages } {
20     \RequirePackage { linguistix-languages }
21 }
22 \IfPackageLoadedF { linguistix-logos } {
23     \RequirePackage { linguistix-logos }
24 }
25 \IfPackageLoadedF { linguistix-nfss } {
26     \RequirePackage { linguistix-nfss }
27 }
28 </package>
```

Set the essentials of the package.

```

29 <*base>
30 \ProvidesExplPackage{linguistix-base}
31     {2025-12-22}
32     {v0.6a}
33     {%
34         The base package of the ‘LinguisTiX’
35         bundle.%
36     }
```

\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.

```

37
38 \cs_new_protected:Npn \lngx_set_keys:n #1 {
39     \keys_set:nn { lngx_keys } { #1 }
40 }
```

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

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

```

41
42 \cs_gset_eq:NN \linguistix \lngx_set_keys:n
43 </base>
```

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

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

```

44 \fixpex
45 \ProvidesExplPackage{linguistix-fixpex}
46     {2025-12-22}
47     {v0.6a}
48     {%
49         The base package of the 'LinguisticTiX'
50         bundle.%
51     }

```

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

```

52
53 \msg_new:nnn { fixpex } { pkg_not_loaded } {
54     The 'LinguisticTiX-fixpex' package is a first-aid~
55     for resolving the conflict between 'unicode-math'~
56     and\ 'expex'. It should only be used if at least~
57     one of the two is loaded. Here~
58     'LinguisticTiX-fixpex' can\ be omitted since you are~
59     not using '1'.
60 }

```

I first start the hook `begindocument/before`.

```

61
62 \hook_gput_code:nnn { begindocument / before } { . } {

```

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

```

63     \IfPackageLoadedTF { expex } {
64         \IfPackageLoadedTF { unicode-math } {
65             \hook_gput_code:nnn { begindocument / end } { . } {

```

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

```

66         \cs_gset_eq:NN \umgla \gla

```

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

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

```

67         \cs_gset_eq:NN \gla \glw@gla
68     }

```

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

```

69     } {
70         \msg_info:nnn { fixpex } { pkg_not_loaded } {
71             unicode-math
72         }
73     }

```

Similarly, I do it for expex.

```
74   } {  
75     \msg_info:nnn { fixpex } { pkg_not_loaded } {  
76       expex  
77     }  
78   }  
79 }  
80 </fixpex>
```

Package essentials first.

```

81 <*font>
82 \ProvidesExplPackage{linguistix-fonts}
83         {2025-12-22}
84         {v0.6a}
85         {%
86         The font-assistant package of the
87         'LinguisTiX' bundle.%
88         }

```

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

```

89
90 \IfPackageLoadedF { linguistix-base } {
91   \RequirePackage { linguistix-base }
92 }
93
94 \IfPackageLoadedF { unicode-math } {
95   \RequirePackage { unicode-math }
96 }
97
98 \IfPackageLoadedF { linguistix-fixpex } {
99   \RequirePackage { linguistix-fixpex }
100 }

```

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

```

101
102 \NewCommandCopy \ogLaTeX \LaTeX
103
104 \RenewDocumentCommand \LaTeX { } {%
105   L\kern-.81ex\relax
106   \raisebox{.6ex}{\textsc{a}}\kern-.23ex\relax
107   \hbox{T}\kern-.4ex\relax
108   \raisebox{-.5ex}{E}\kern-.3ex\relax
109   X%
110 }

```

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

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

```

111
112 \keys_define:nn { lngx_keys } {
113   old~ style~ numbers
114   .bool_gset:N          = {
115     \g_lngx_old_style_bool
116   },
117   old~ style~ one
118   .bool_gset:N          = {
119     \g_lngx_old_style_one_bool
120   },
121   bourbaki's~ empty~ set
122   .bool_gset:N          = {
123     \g_lngx_bourbaki_bool
124   }

```

125 }

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

```
\g__lngx_text_fonts_prop
  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 bold swash features
  text small caps
text small caps features
```

In the first few versions of the package, I used to save the font-names and their features in token lists, but I found a better way to deal with this later which was using `prop` lists. I had released the `tl`s publicly (with a single `_` after the scope marker), which means ideally they should be available forever, but for performance and maintenance the newer approach is much preferred and hence I decided to shift to `prop` lists from v0.6. This time, I am correcting the mistake I made before. The `prop` list that saves the keys is not public. It need not be. Only the key-value pairs are public. They are unchanged anyway. This section describes the implementation of 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>.)

```
126
127 \prop_gclear_new:N \g__lngx_text_fonts_prop
128
129 \clist_map_inline:nn {
130   upright,
131   bold~ upright,
132   italic,
133   bold~ italic,
134   slanted,
135   bold~ slanted,
136   swash,
137   bold~ swash,
138   small~ caps
139 } {
```

All the keys here are prefixed with the word `text` in order to distinguish them from the keys provided by the `LINGUISTX-IPA` package. The argument of these keys should be expanded for which I use `\prop_gput:Nne` function. Each `#1` is replaced by the items from `clist` and the loop is repeated, whereas `##1` is the argument passed to the key by user.

```
140 \keys_define:nn { lngx_keys } {
141   text~ #1
142   .code:n = {
143     \prop_gput:Nne \g__lngx_text_fonts_prop
144       { text~ #1 }
145     { ##1 }
146   },
147   text~ #1~ features
148   .code:n = {
149     \group_begin:
150     \tl_clear:N \l_tmpa_tl
151     \tl_clear:N \l_tmpb_tl
152     \prop_get:NeN \g__lngx_text_fonts_prop
153       { text~ #1~ features }
154     \l_tmpa_tl
155     \quark_if_no_value:NTF \l_tmpa_tl {
156       \tl_set:Ne \l_tmpb_tl { ##1 }
157     } {
```



```

158         \tl_set:Nc \l_tmpb_tl { \l_tmpa_tl, ##1 }
159     }
160     \prop_gput:NnV \g__lngx_text_fonts_prop
161         { text~ #1~ features }
162         \l_tmpb_tl
163     \group_end:
164 }
165 }
166 }

```

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

text extra features This key adds to the property that stores the extra features for the serif fonts.

```

167
168 \keys_define:nn { lngx_keys } {
169     text~ extra~ features
170     .code:n          = {
171         \group_begin:
172         \tl_clear:N \l_tmpa_tl
173         \tl_clear:N \l_tmpb_tl
174         \prop_get:NeN \g__lngx_text_fonts_prop
175             { text~ extra~ features }
176             \l_tmpa_tl
177         \quark_if_no_value:NTF \l_tmpa_tl {
178             \tl_set:Nc \l_tmpb_tl { #1 }
179         } {
180             \tl_set:Nc \l_tmpb_tl { \l_tmpa_tl, #1 }
181         }
182         \prop_gput:NnV \g__lngx_text_fonts_prop
183             { text~ extra~ features }
184             \l_tmpb_tl
185         \group_end:
186     }
187 }

```

(End of definition for `text extra features`. This function is documented on page 9.)

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
 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

Since the only difference between the upcoming keys is that of the word **sans** and **mono**, we combine them together and use a nested **clist**. The rest of the mechanism is identical.

```

188
189 \clist_map_inline:nn {
190   sans,
191   mono
192 } {
193   \clist_map_inline:nn {
194     upright,
195     bold~ upright,
196     italic,
197     bold~ italic,
198     slanted,
199     bold~ slanted,
200     swash,
201     bold~ swash,
202     small~ caps
203   } {
204     \keys_define:nn { lngx_keys } {
205       text~ #1~ ##1
206       .code:n          = {
207         \prop_gput:Nne \g__lngx_text_fonts_prop
208           { text~ #1~ ##1 }
209           { #####1 }
210       },
211       text~ #1~ ##1~ features
212       .code:n          = {
213         \group_begin:
214         \tl_clear:N \l_tmpa_tl
215         \tl_clear:N \l_tmpb_tl
216         \prop_get:Nen \g__lngx_text_fonts_prop
217           { text #1~ ##1~ features }
218           \l_tmpa_tl
219         \quark_if_no_value:NTF \l_tmpa_tl {
220           \tl_set:Nen \l_tmpb_tl { #####1 }
221         } {
222           \tl_set:Nen \l_tmpb_tl { \l_tmpa_tl, #####1 }
223         }
224         \prop_gput:NnV \g__lngx_text_fonts_prop
225           { text~ #1~ ##1~ features }
226           \l_tmpb_tl
227         \group_end:
228       }
229     }
230   }
231   \keys_define:nn { lngx_keys } {
232     text~ #1~ extra~ features
233     .code:n          = {
234       \group_begin:
235       \tl_clear:N \l_tmpa_tl
236       \tl_clear:N \l_tmpb_tl
237       \prop_get:Nen \g__lngx_text_fonts_prop
238         { text~ #1~ extra~ features }
239         \l_tmpa_tl

```

```

240     \quark_if_no_value:NTF \l_tmpa_tl {
241       \tl_set:Ne \l_tmpb_tl { ##1 }
242     } {
243       \tl_set:Ne \l_tmpb_tl { \l_tmpa_tl, ##1 }
244     }
245     \prop_gput:NnV \g__lngx_text_fonts_prop
246                 { text~ #1~ extra~ features }
247                 \l_tmpb_tl
248   \group_end:
249 }
250 }
251 }

```

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

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

```

math features
math bold
math bold features
252 \keys_define:nn { lngx_keys } {
253   math
254   .code:n          = {
255     \prop_gput:Nne \g__lngx_text_fonts_prop
256                 { math }
257                 { #1 }
258   },
259   math~ features
260   .code:n          = {
261     \group_begin:
262     \tl_clear:N \l_tmpa_tl
263     \tl_clear:N \l_tmpb_tl
264     \prop_get:NeN \g__lngx_text_fonts_prop
265                 { math~ features }
266                 \l_tmpa_tl
267     \quark_if_no_value:NTF \l_tmpa_tl {
268       \tl_set:Ne \l_tmpb_tl { #1 }
269     } {
270       \tl_set:Ne \l_tmpb_tl { \l_tmpa_tl, #1 }
271     }
272     \prop_gput:NnV \g__lngx_text_fonts_prop
273                 { math~ features }
274                 \l_tmpb_tl
275     \group_end:
276   },
277   math~ bold
278   .code:n          = {
279     \prop_gput:Nne \g__lngx_text_fonts_prop
280                 { math~ bold }
281                 { #1 }
282   },
283   .code:n          = {
284     \group_begin:
285     \tl_clear:N \l_tmpa_tl
286     \tl_clear:N \l_tmpb_tl
287     \prop_get:NeN \g__lngx_text_fonts_prop
288                 { math~ bold~ features }
289

```

```

290         \l_tmpa_tl
291     \quark_if_no_value:NTF \l_tmpa_tl {
292         \tl_set:Nc \l_tmpb_tl { #1 }
293     } {
294         \tl_set:Nc \l_tmpb_tl { \l_tmpa_tl, #1 }
295     }
296     \prop_gput:NnV \g__lngx_text_fonts_prop
297                 { math~ bold~ features }
298                 \l_tmpb_tl
299     \group_end:
300 }
301 }

```

(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 book weight and in all of the serif, sans serif and monospaced families. 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`.

```

302
303 \keys_define:nn { lngx_keys } {
304     newcm
305     .meta:n = {
306         text~ extra~
307         features = {IgnoreFontspecFile},
308         text~ sans~ extra~
309         features = {IgnoreFontspecFile},
310         text~ mono~ extra~
311         features = {IgnoreFontspecFile},
312         text~
313         upright = {NewCM10-Book.otf},
314         text~
315         bold~ upright = {NewCM10-Bold.otf},
316         text~
317         italic = {NewCM10-BookItalic.otf},
318         text~
319         bold~ italic = {NewCM10-BoldItalic.otf},
320         text~
321         slanted = {NewCM10-Book.otf},
322         text~
323         bold~ slanted = {NewCM10-Bold.otf},
324         text~
325         swash = {NewCM10-Book.otf},
326         text~
327         bold~ swash = {NewCM10-Bold.otf},
328         text~ sans~
329         upright = {NewCMSans10-Book.otf},
330         text~ sans~
331         bold~ upright = {NewCMSans10-Bold.otf},
332         text~ sans~
333         italic = {NewCMSans10-BookOblique.otf},
334         text~ sans~
335         bold~ italic = {NewCMSans10-BoldOblique.otf},
336         text~ sans~

```

```

337     slanted                = {NewCMSans10-Book.otf},
338     text~ sans~           = {NewCMSans10-Bold.otf},
339     bold~ slanted         = {NewCMSans10-Bold.otf},
340     text~ sans~           = {NewCMSans10-Book.otf},
341     swash                  = {NewCMSans10-Book.otf},
342     text~ sans~           = {NewCMSans10-Bold.otf},
343     bold~ swash           = {NewCMSans10-Bold.otf},
344     text~ mono~           = {NewCMMono10-Book.otf},
345     upright                = {NewCMMono10-Book.otf},
346     text~ mono~           = {NewCMMono10-Bold.otf},
347     bold~ upright         = {NewCMMono10-Bold.otf},
348     text~ mono~           = {NewCMMono10-BookItalic.otf},
349     italic                 = {NewCMMono10-BookItalic.otf},
350     text~ mono~           = {NewCMMono10-BoldOblique.otf},
351     bold~ italic          = {NewCMMono10-BoldOblique.otf},
352     text~ mono~           = {NewCMMono10-Book.otf},
353     slanted                = {NewCMMono10-Book.otf},
354     text~ mono~           = {NewCMMono10-Bold.otf},
355     bold~ slanted         = {NewCMMono10-Bold.otf},
356     text~ mono~           = {NewCMMono10-Book.otf},
357     swash                  = {NewCMMono10-Book.otf},
358     text~ mono~           = {NewCMMono10-Bold.otf},
359     bold~ swash           = {NewCMMono10-Bold.otf},
360     math                   = {NewCMMath-Book.otf},
361     math~ bold             = {NewCMMath-Bold.otf}
362   }
363 }

```

(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.

```

364 \keys_define:nn { lngx_keys } {
365   newcm~ sans
366   .meta:n
367   = {
368     text~ extra~
369     features          = {IgnoreFontspecFile},
370     text~
371     upright           = {NewCMSans10-Book.otf},
372     text~
373     bold~ upright     = {NewCMSans10-Bold.otf},
374     text~
375     italic            = {NewCMSans10-BookOblique.otf},
376     text~
377     bold~ italic      = {NewCMSans10-BoldOblique.otf},
378     text~
379     slanted           = {NewCMSans10-Book.otf},
380     text~
381     bold~ slanted     = {NewCMSans10-Bold.otf},
382     text~
383     swash             = {NewCMSans10-Book.otf},
384     text~
385     bold~ swash       = {NewCMSans10-Bold.otf},
386     math              = {NewCMSansMath-Regular.otf},

```

```

387   math~ bold           = {NewCMSansMath-Regular.otf}
388 }
389 }

```

(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.

```

390
391 \keys_define:nn { lngx_keys } {
392   newcm~ mono
393   .meta:n           = {
394     text~
395     upright          = {NewCMMono10-Book.otf},
396     text~
397     bold~ upright    = {NewCMMono10-Bold.otf},
398     text~
399     italic           = {NewCMMono10-BookItalic.otf},
400     text~
401     bold~ italic     = {NewCMMono10-BoldOblique.otf},
402     text~
403     slanted          = {NewCMMono10-Book.otf},
404     text~
405     bold~ slanted    = {NewCMMono10-Bold.otf},
406     text~
407     swash            = {NewCMMono10-Book.otf},
408     text~
409     bold~ swash      = {NewCMMono10-Bold.otf},
410     math              = {NewCMSansMath-Regular.otf},
411     math~ bold       = {NewCMSansMath-Regular.otf}
412   }
413 }

```

(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.

```

414
415 \keys_define:nn { lngx_keys } {
416   newcm~ regular
417   .meta:n           = {
418     text~ extra~
419     features          = {IgnoreFontspecFile},
420     text~ sans~ extra~
421     features          = {IgnoreFontspecFile},
422     text~
423     upright           = {NewCM10-Regular.otf},
424     text~
425     bold~ upright     = {NewCM10-Bold.otf},
426     text~
427     italic            = {NewCM10-Italic.otf},
428     text~
429     bold~ italic      = {NewCM10-BoldItalic.otf},
430     text~
431     slanted           = {NewCM10-Regular.otf},

```

```

432 text~
433 bold~ slanted = {NewCM10-Bold.otf},
434 text~
435 swash = {NewCM10-Regular.otf},
436 text~
437 bold~ swash = {NewCM10-Bold.otf},
438 text~ sans~
439 upright = {NewCMSans10-Regular.otf},
440 text~ sans~
441 bold~ upright = {NewCMSans10-Bold.otf},
442 text~ sans~
443 italic = {NewCMSans10-Oblique.otf},
444 text~ sans~
445 bold~ italic = {NewCMSans10-BoldOblique.otf},
446 text~ sans~
447 slanted = {NewCMSans10-Regular.otf},
448 text~ sans~
449 bold~ slanted = {NewCMSans10-Bold.otf},
450 text~ sans~
451 swash = {NewCMSans10-Regular.otf},
452 text~ sans~
453 bold~ swash = {NewCMSans10-Bold.otf},
454 text~ mono~
455 upright = {NewCMMono10-Regular.otf},
456 text~ mono~
457 bold~ upright = {NewCMMono10-Bold.otf},
458 text~ mono~
459 italic = {NewCMMono10-Italic.otf},
460 text~ mono~
461 bold~ italic = {NewCMMono10-BoldOblique.otf},
462 text~ mono~
463 slanted = {NewCMMono10-Regular.otf},
464 text~ mono~
465 bold~ slanted = {NewCMMono10-Bold.otf},
466 text~ mono~
467 swash = {NewCMMono10-Regular.otf},
468 text~ mono~
469 bold~ swash = {NewCMMono10-Bold.otf},
470 math = {NewCMMath-Regular.otf},
471 math~ bold = {NewCMMath-Bold.otf}
472 }
473 }

```

(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.

```

474
475 \keys_define:nn { lngx_keys } {
476   newcm~ regular~ sans
477   .meta:n = {
478     text~ extra~
479     features = {IgnoreFontspecFile},
480     text~

```

```

481    upright                = {NewCMSans10-Regular.otf},
482    text~
483    bold~ upright          = {NewCMSans10-Bold.otf},
484    text~
485    italic                 = {NewCMSans10-Oblique.otf},
486    text~
487    bold~ italic           = {NewCMSans10-BoldOblique.otf},
488    text~
489    slanted                = {NewCMSans10-Regular.otf},
490    text~
491    bold~ slanted          = {NewCMSans10-Bold.otf},
492    text~
493    swash                  = {NewCMSans10-Regular.otf},
494    text~
495    bold~ swash            = {NewCMSans10-Bold.otf},
496    math                   = {NewCMSansMath-Regular.otf},
497    math~ bold             = {NewCMSansMath-Regular.otf},
498    math                   = {NewCMSansMath-Regular.otf},
499    math~ bold             = {NewCMSansMath-Regular.otf}
500  }
501 }

```

(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.

```

502
503 \keys_define:nn { lngx_keys } {
504   newcm~ regular~ mono
505   .meta:n                = {
506     text~
507     upright                = {NewCMMono10-Regular.otf},
508     text~
509     bold~ upright          = {NewCMMono10-Bold.otf},
510     text~
511     italic                 = {NewCMMono10-Italic.otf},
512     text~
513     bold~ italic           = {NewCMMono10-BoldOblique.otf},
514     text~
515     slanted                = {NewCMMono10-Regular.otf},
516     text~
517     bold~ slanted          = {NewCMMono10-Bold.otf},
518     text~
519     swash                  = {NewCMMono10-Regular.otf},
520     text~
521     bold~ swash            = {NewCMMono10-Bold.otf},
522     math                   = {NewCMSansMath-Regular.otf},
523     math~ bold             = {NewCMSansMath-Regular.otf}
524   }
525 }

```

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

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


```

526
527 \lngx_set_keys:n {
528   bourbaki's~ empty~ set,

```

Then we load the old style numbers boolean.

```

529   old~ style~ numbers,
530   newcm
531 }

```

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

```

532
533 \sys_if_engine luatex:T {
534   \lngx_set_keys:n {
535     text~ extra~
536     features          = {
537       Renderer        = { HarfBuzz }
538     },
539     text~ sans~ extra~
540     features          = {
541       Renderer        = { HarfBuzz }
542     },
543     text~ mono~ extra~
544     features          = {
545       Renderer        = { HarfBuzz }
546     }
547   }
548 }

```

`\lngx_set_main_font:nn` If LINGUISTIX-LANGUAGES package is loaded, I load the fonts with `\babelfont` command.
`\lngx_set_sans_font:nn` In case it is not loaded, the fonts are set with `\setxxxxcommand`-type commands provided
`\lngx_set_mono_font:nn` by fontspec.
`\lngx_set_math_font:nn`

```

549
550 \IfPackageLoadedF { linguistix-languages } {
551   \cs_new_protected:Npn \lngx_set_main_font:nn #1#2 {
552     \setmainfont [ #1 ] { #2 }
553   }
554   \cs_new_protected:Npn \lngx_set_sans_font:nn #1#2 {
555     \setsansfont [ #1 ] { #2 }
556   }
557   \cs_new_protected:Npn \lngx_set_mono_font:nn #1#2 {
558     \setmonofont [ #1 ] { #2 }
559   }
560 }

```

A wrapper command is provided for loading math fonts.

```

561
562 \cs_new_protected:Npn \lngx_set_math_font:nn #1#2 {
563   \setmathfont [ #1 ] { #2 }
564 }

```

All of these commands should expand their arguments, so I provide the appropriate variants.

```

565
566 \cs_generate_variant:Nn \lngx_set_main_font:nn { ee }

```

```

567 \cs_generate_variant:Nn \lngx_set_sans_font:nn { ee }
568 \cs_generate_variant:Nn \lngx_set_mono_font:nn { ee }
569 \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 15.)

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.

```

570
571 \hook_gput_code:nnn { begindocument / before } { . } {
572   \lngx_set_keys:n {
573     text~ extra~
574     features          = {
575       \bool_if:NT \g_lngx_old_style_bool {
576         Numbers          = { OldStyle },
577         \bool_if:NT \g_lngx_old_style_one_bool {
578           CharacterVariant = { 6 }
579         }
580       }
581     },
582     text~ sans~ extra~
583     features          = {
584       \bool_if:NT \g_lngx_old_style_bool {
585         Numbers          = { OldStyle },
586         \bool_if:NT \g_lngx_old_style_one_bool {
587           CharacterVariant = { 6 }
588         }
589       }
590     }
591   }
592   \group_begin:
593   \clist_map_inline:nn {
594     upright,
595     slanted,
596     swash
597   } {
598     \prop_get:NnN \g__lngx_text_fonts_prop
599       { text~ #1 }
600       \l_tmpa_tl
601     \tl_if_eq:NnT \l_tmpa_tl { NewCM10-Book.otf } {
602       \lngx_set_keys:n {
603         text~ #1~
604         features          = {
605           SizeFeatures    = {
606             {
607               Size          = {-8},
608               Font          = {
609                 NewCM08-Book.otf
610             }
611           },
612           {
613             Size          = {8-},

```

```

614         Font = {
615             NewCM10-Book.otf
616         }
617     }
618 }
619 }
620 }
621 }
622 \tl_if_eq:NnT \l_tmpa_tl { NewCM10-Regular.otf } {
623     \lngx_set_keys:n {
624         text~ #1~
625         features = {
626             SizeFeatures = {
627                 {
628                     Size = {-8},
629                     Font = {
630                         NewCM08-Regular.otf
631                     }
632                 },
633                 {
634                     Size = {8-},
635                     Font = {
636                         NewCM10-Regular.otf
637                     }
638                 }
639             }
640         }
641     }
642 }
643 \tl_if_eq:NnT \l_tmpa_tl { NewCMSans10-Book.otf } {
644     \lngx_set_keys:n {
645         text~ #1~
646         features = {
647             SizeFeatures = {
648                 {
649                     Size = {-8},
650                     Font = {
651                         NewCMSans08-Book.otf
652                     }
653                 },
654                 {
655                     Size = {8-},
656                     Font = {
657                         NewCMSans10-Book.otf
658                     }
659                 }
660             }
661         }
662     }
663 }
664 \tl_if_eq:NnT \l_tmpa_tl { NewCMSans10-Regular.otf } {
665     \lngx_set_keys:n {
666         text~ #1~
667         features = {

```

```

668         SizeFeatures          = {
669             {
670                 Size            = {-8},
671                 Font            = {
672                     NewCMSans08-Regular.otf
673                 }
674             },
675             {
676                 Size            = {8-},
677                 Font            = {
678                     NewCMSans10-Regular.otf
679                 }
680             }
681         }
682     }
683 }
684 }
685 \tl_if_eq:NnT \l_tmpa_tl { NewCMMono10-Book.otf } {
686     \lngx_set_keys:n {
687         text~ #1~
688         features          = {
689             SizeFeatures  = {
690                 {
691                     Size            = {0-},
692                     Font            = {
693                         NewCMMono10-Book.otf
694                     }
695                 }
696             }
697         }
698     }
699 }
700 \tl_if_eq:NnT \l_tmpa_tl { NewCMMono10-Regular.otf } {
701     \lngx_set_keys:n {
702         text~ #1~
703         features          = {
704             SizeFeatures  = {
705                 {
706                     Size            = {0-},
707                     Font            = {
708                         NewCMMono10-Regular.otf
709                     }
710                 }
711             }
712         }
713     }
714 }
715 \prop_get:NnN \g__lngx_text_fonts_prop
716     { text~ sans~ #1 }
717     \l_tmpa_tl
718 \tl_if_eq:NnT \l_tmpa_tl { NewCMSans10-Book.otf } {
719     \lngx_set_keys:n {
720         text~ sans~ #1~
721         features          = {

```

```

722         SizeFeatures          = {
723             {
724                 Size            = {-8},
725                 Font            = {
726                     NewCMSans08-Book.otf
727                 }
728             },
729             {
730                 Size            = {8-},
731                 Font            = {
732                     NewCMSans10-Book.otf
733                 }
734             }
735         }
736     }
737 }
738 }
739 \tl_if_eq:NnT \l_tmpa_tl { NewCMSans10-Regular.otf } {
740     \lngx_set_keys:n {
741         text~ sans~ #1~
742         features          = {
743             SizeFeatures  = {
744                 {
745                     Size            = {-8},
746                     Font            = {
747                         NewCMSans08-Regular.otf
748                     }
749                 },
750                 {
751                     Size            = {8-},
752                     Font            = {
753                         NewCMSans10-Regular.otf
754                     }
755                 }
756             }
757         }
758     }
759 }
760 \prop_get:NnN \g__lngx_text_fonts_prop
761     { text~ mono~ #1 }
762     \l_tmpa_tl
763 \tl_if_eq:NnT \l_tmpa_tl { NewCMMono10-Book.otf } {
764     \lngx_set_keys:n {
765         text~ mono~ #1~
766         features          = {
767             SizeFeatures  = {
768                 {
769                     Size            = {0-},
770                     Font            = {
771                         NewCMMono10-Book.otf
772                     }
773                 }
774             }
775         }

```

```

776     }
777   }
778   \tl_if_eq:NnT \l_tmpa_tl { NewCMMono10-Regular.otf } {
779     \lngx_set_keys:n {
780       text~ mono~ #1~
781       features          = {
782         SizeFeatures    = {
783           {
784             Size        = {0-},
785             Font        = {
786               NewCMMono10-Regular.otf
787             }
788           }
789         }
790       }
791     }
792   }
793 }
794 \clist_map_inline:nn {
795   upright,
796   slanted,
797   swash
798 } {
799   \prop_get:NnN \g__lngx_text_fonts_prop
800   { text~ bold~ #1 }
801   \l_tmpa_tl
802   \tl_if_eq:NnT \l_tmpa_tl { NewCM10-Bold.otf } {
803     \lngx_set_keys:n {
804       text~ bold~ #1~
805       features          = {
806         SizeFeatures    = {
807           {
808             Size        = {0-},
809             Font        = {
810               NewCM10-Bold.otf
811             }
812           }
813         }
814       }
815     }
816   }
817   \tl_if_eq:NnT \l_tmpa_tl { NewCMSans10-Bold.otf } {
818     \lngx_set_keys:n {
819       text~ bold~ #1~
820       features          = {
821         {
822           Size          = {0-},
823           Font          = {
824             NewCMSans10-Bold.otf
825           }
826         }
827       }
828     }
829   }

```

```

830 \tl_if_eq:NnT \l_tmpa_tl { NewCMMono10-Bold.otf } {
831   \lngx_set_keys:n {
832     text~ bold~ #1~
833     features = {
834       SizeFeatures = {
835         {
836           Size = {0-},
837           Font = {
838             NewCMMono10-Bold.otf
839           }
840         }
841       }
842     }
843   }
844 }
845 \prop_get:NnN \g__lngx_text_fonts_prop
846   { text~ sans~ bold~ #1 }
847   \l_tmpa_tl
848 \tl_if_eq:NnT \l_tmpa_tl { NewCMSans10-Bold.otf } {
849   \lngx_set_keys:n {
850     text~ sans~ bold~ #1~
851     features = {
852       SizeFeatures = {
853         {
854           Size = {0-},
855           Font = {
856             NewCMSans10-Bold.otf
857           }
858         }
859       }
860     }
861   }
862 }
863 \prop_get:NnN \g__lngx_text_fonts_prop
864   { text~ mono~ bold~ #1 }
865   \l_tmpa_tl
866 \tl_if_eq:NnT \l_tmpa_tl { NewCMMono10-Bold.otf } {
867   \lngx_set_keys:n {
868     text~ mono~ bold~ #1~
869     features = {
870       SizeFeatures = {
871         {
872           Size = {0-},
873           Font = {
874             NewCMMono10-Bold.otf
875           }
876         }
877       }
878     }
879   }
880 }
881 }
882 \prop_get:NnN \g__lngx_text_fonts_prop
883   { text~ italic }

```

```

884         \l_tmpa_tl
885 \tl_if_eq:NnT \l_tmpa_tl { NewCM10-BookItalic.otf } {
886     \lngx_set_keys:n {
887         text~ italic~
888         features = {
889             SizeFeatures = {
890                 {
891                     Size = {-8},
892                     Font = {
893                         NewCM08-BookItalic.otf
894                     }
895                 },
896                 {
897                     Size = {8-},
898                     Font = {
899                         NewCM10-BookItalic.otf
900                     }
901                 }
902             }
903         }
904     }
905 }
906 \tl_if_eq:NnT \l_tmpa_tl { NewCM10-Italic.otf } {
907     \lngx_set_keys:n {
908         text~ italic~
909         features = {
910             SizeFeatures = {
911                 {
912                     Size = {-8},
913                     Font = {
914                         NewCM08-Italic.otf
915                     }
916                 },
917                 {
918                     Size = {8-},
919                     Font = {
920                         NewCM10-Italic.otf
921                     }
922                 }
923             }
924         }
925     }
926 }
927 \tl_if_eq:NnT \l_tmpa_tl { NewCMSans10-BookOblique.otf } {
928     \lngx_set_keys:n {
929         text~ italic~
930         features = {
931             SizeFeatures = {
932                 {
933                     Size = {-8},
934                     Font = {
935                         NewCMSans08-BookOblique.otf
936                     }
937                 },

```



```

938         {
939             Size                = {8-},
940             Font                 = {
941                 NewCMSans10-BookOblique.otf
942             }
943         }
944     }
945 }
946 }
947 }
948 \tl_if_eq:NnT \l_tmpa_tl { NewCMSans10-Oblique.otf } {
949     \lngx_set_keys:n {
950         text~ italic~
951         features                = {
952             SizeFeatures        = {
953                 {
954                     Size        = {-8},
955                     Font        = {
956                         NewCMSans08-Oblique.otf
957                     }
958                 },
959                 {
960                     Size        = {8-},
961                     Font        = {
962                         NewCMSans10-Oblique.otf
963                     }
964                 }
965             }
966         }
967     }
968 }
969 \tl_if_eq:NnT \l_tmpa_tl { NewCMMono10-BookItalic.otf } {
970     \lngx_set_keys:n {
971         text~ italic~
972         features                = {
973             SizeFeatures        = {
974                 {
975                     Size        = {0-},
976                     Font        = {
977                         NewCMMono10-BookItalic.otf
978                     }
979                 }
980             }
981         }
982     }
983 }
984 \tl_if_eq:NnT \l_tmpa_tl { NewCMMono10-Italic.otf } {
985     \lngx_set_keys:n {
986         text~ italic~
987         features                = {
988             SizeFeatures        = {
989                 {
990                     Size        = {0-},
991                     Font        = {

```

```

992             NewCMMono10-Italic.otf
993         }
994     }
995 }
996 }
997 }
998 }
999 \prop_get:NnN \g__lmgx_text_fonts_prop
1000     { text~ sans~ italic }
1001     \l_tmpa_tl
1002 \tl_if_eq:NnT \l_tmpa_tl { NewCMSans10-BookOblique.otf } {
1003     \lmgx_set_keys:n {
1004         text~ sans~ italic~
1005         features = {
1006             SizeFeatures = {
1007                 {
1008                     Size = {-8},
1009                     Font = {
1010                         NewCMSans08-BookOblique.otf
1011                     }
1012                 },
1013                 {
1014                     Size = {8-},
1015                     Font = {
1016                         NewCMSans10-BookOblique.otf
1017                     }
1018                 }
1019             }
1020         }
1021     }
1022 }
1023 \tl_if_eq:NnT \l_tmpa_tl { NewCMSans10-Oblique.otf } {
1024     \lmgx_set_keys:n {
1025         text~ sans~ italic~
1026         features = {
1027             SizeFeatures = {
1028                 {
1029                     Size = {-8},
1030                     Font = {
1031                         NewCMSans08-Oblique.otf
1032                     }
1033                 },
1034                 {
1035                     Size = {8-},
1036                     Font = {
1037                         NewCMSans10-Oblique.otf
1038                     }
1039                 }
1040             }
1041         }
1042     }
1043 }
1044 \prop_get:NnN \g__lmgx_text_fonts_prop
1045     { text~ mono~ italic }

```

```

1046         \l_tmpa_tl
1047 \tl_if_eq:NnT \l_tmpa_tl { NewCMMono10-BookItalic.otf } {
1048   \lngx_set_keys:n {
1049     text~ mono~ italic~
1050     features = {
1051       SizeFeatures = {
1052         {
1053           Size = {0-},
1054           Font = {
1055             NewCMMono10-BookItalic.otf
1056           }
1057         }
1058       }
1059     }
1060   }
1061 }
1062 \tl_if_eq:NnT \l_tmpa_tl { NewCMMono10-Italic.otf } {
1063   \lngx_set_keys:n {
1064     text~ mono~ italic~
1065     features = {
1066       SizeFeatures = {
1067         {
1068           Size = {0-},
1069           Font = {
1070             NewCMMono10-Italic.otf
1071           }
1072         }
1073       }
1074     }
1075   }
1076 }
1077 \prop_get:NnN \g__lngx_text_fonts_prop
1078   { text~ bold~ italic }
1079   \l_tmpa_tl
1080 \tl_if_eq:NnT \l_tmpa_tl { NewCM10-BoldItalic.otf } {
1081   \lngx_set_keys:n {
1082     text~ bold~ italic~
1083     features = {
1084       SizeFeatures = {
1085         {
1086           Size = {0-},
1087           Font = {
1088             NewCM10-BoldItalic.otf
1089           }
1090         }
1091       }
1092     }
1093   }
1094 }
1095 \tl_if_eq:NnT \l_tmpa_tl { NewCMSans10-BoldOblique.otf } {
1096   \lngx_set_keys:n {
1097     text~ sans~ bold~ italic~
1098     features = {
1099       SizeFeatures = {

```

```

II00         {
II01             Size                      = {0-},
II02             Font                      = {
II03                 NewCMSans10-BoldOblique.otf
II04             }
II05         }
II06     }
II07 }
II08 }
II09 }
II10 \tl_if_eq:NnT \l_tmpa_tl { NewCMMono10-BoldOblique.otf } {
II11     \lngx_set_keys:n {
II12         text~ mono~ bold~ italic~
II13         features                      = {
II14             SizeFeatures              = {
II15                 {
II16                     Size              = {0-},
II17                     Font              = {
II18                         NewCMMono10-BoldOblique.otf
II19                     }
II20                 }
II21             }
II22         }
II23     }
II24 }
II25 \prop_get:NnN \g__lngx_text_fonts_prop
II26             { text~ sans~ bold~ italic }
II27             \l_tmpa_tl
II28 \tl_if_eq:NnT \l_tmpa_tl { NewCMSans10-BoldOblique.otf } {
II29     \lngx_set_keys:n {
II30         text~ sans~ bold~ italic~
II31         features                      = {
II32             SizeFeatures              = {
II33                 {
II34                     Size              = {0-},
II35                     Font              = {
II36                         NewCMSans10-BoldOblique.otf
II37                     }
II38                 }
II39             }
II40         }
II41     }
II42 }
II43 \prop_get:NnN \g__lngx_text_fonts_prop
II44             { text~ mono~ bold~ italic }
II45             \l_tmpa_tl
II46 \tl_if_eq:NnT \l_tmpa_tl { NewCMMono10-BoldOblique.otf } {
II47     \lngx_set_keys:n {
II48         text~ mono~ bold~ italic~
II49         features                      = {
II50             SizeFeatures              = {
II51                 {
II52                     Size              = {0-},
II53                     Font              = {

```

```

1154         NewCMMono10-BoldOblique.otf
1155     }
1156 }
1157 }
1158 }
1159 }
1160 }
1161 \group_end:
1162 \lngx_set_main_font:ee {
1163     \prop_item:Nn \g__lngx_text_fonts_prop {
1164         text~ extra~ features
1165     },
1166     UprightFont = {
1167         \prop_item:Nn \g__lngx_text_fonts_prop {
1168             text~ upright
1169         }
1170     },
1171     UprightFeatures = {
1172         \prop_item:Nn \g__lngx_text_fonts_prop {
1173             text~ upright~ features
1174         }
1175     },
1176     ItalicFont = {
1177         \prop_item:Nn \g__lngx_text_fonts_prop {
1178             text~ italic
1179         }
1180     },
1181     ItalicFeatures = {
1182         \prop_item:Nn \g__lngx_text_fonts_prop {
1183             text~ italic~ features
1184         }
1185     },
1186     BoldFont = {
1187         \prop_item:Nn \g__lngx_text_fonts_prop {
1188             text~ bold~ upright
1189         }
1190     },
1191     BoldFeatures = {
1192         \prop_item:Nn \g__lngx_text_fonts_prop {
1193             text~ bold~ upright~ features
1194         }
1195     },
1196     BoldItalicFont = {
1197         \prop_item:Nn \g__lngx_text_fonts_prop {
1198             text~ bold~ italic
1199         }
1200     },
1201     BoldItalicFeatures = {
1202         \prop_item:Nn \g__lngx_text_fonts_prop {
1203             text~ bold~ italic~ features
1204         }
1205     },
1206     SlantedFont = {
1207         \prop_item:Nn \g__lngx_text_fonts_prop {

```

```

I208         text~ slanted
I209     }
I210 },
I211 SlantedFeatures = {
I212     \prop_item:Nn \g__lngx_text_fonts_prop {
I213         text~ slanted~ features
I214     }
I215 },
I216 BoldSlantedFont = {
I217     \prop_item:Nn \g__lngx_text_fonts_prop {
I218         text~ bold~ slanted
I219     }
I220 },
I221 BoldSlantedFeatures = {
I222     \prop_item:Nn \g__lngx_text_fonts_prop {
I223         text~ bold~ slanted~ features
I224     }
I225 },
I226 SwashFont = {
I227     \prop_item:Nn \g__lngx_text_fonts_prop {
I228         text~ swash
I229     }
I230 },
I231 SwashFeatures = {
I232     \prop_item:Nn \g__lngx_text_fonts_prop {
I233         text~ swash~ features
I234     }
I235 },
I236 BoldSwashFont = {
I237     \prop_item:Nn \g__lngx_text_fonts_prop {
I238         text~ bold~ swash
I239     }
I240 },
I241 BoldSwashFeatures = {
I242     \prop_item:Nn \g__lngx_text_fonts_prop {
I243         text~ bold~ swash~ features
I244     }
I245 }
I246 } {
I247     \prop_item:Nn \g__lngx_text_fonts_prop {
I248         text~ upright
I249     }
I250 }
I251 \lngx_set_sans_font:ee {
I252     \prop_item:Nn \g__lngx_text_fonts_prop {
I253         text~ sans~ extra~ features
I254     },
I255     UprightFont = {
I256         \prop_item:Nn \g__lngx_text_fonts_prop {
I257             text~ sans~ upright
I258         }
I259     },
I260     UprightFeatures = {
I261         \prop_item:Nn \g__lngx_text_fonts_prop {

```

```

1262         text~ sans~ upright~ features
1263     }
1264 },
1265 ItalicFont = {
1266     \prop_item:Nn \g__lngx_text_fonts_prop {
1267         text~ sans~ italic
1268     }
1269 },
1270 ItalicFeatures = {
1271     \prop_item:Nn \g__lngx_text_fonts_prop {
1272         text~ sans~ italic~ features
1273     }
1274 },
1275 BoldFont = {
1276     \prop_item:Nn \g__lngx_text_fonts_prop {
1277         text~ sans~ bold~ upright
1278     }
1279 },
1280 BoldFeatures = {
1281     \prop_item:Nn \g__lngx_text_fonts_prop {
1282         text~ sans~ bold~ upright~ features
1283     }
1284 },
1285 BoldItalicFont = {
1286     \prop_item:Nn \g__lngx_text_fonts_prop {
1287         text~ sans~ bold~ italic
1288     }
1289 },
1290 BoldItalicFeatures = {
1291     \prop_item:Nn \g__lngx_text_fonts_prop {
1292         text~ sans~ bold~ italic~ features
1293     }
1294 },
1295 SlantedFont = {
1296     \prop_item:Nn \g__lngx_text_fonts_prop {
1297         text~ sans~ slanted
1298     }
1299 },
1300 SlantedFeatures = {
1301     \prop_item:Nn \g__lngx_text_fonts_prop {
1302         text~ sans~ slanted~ features
1303     }
1304 },
1305 BoldSlantedFont = {
1306     \prop_item:Nn \g__lngx_text_fonts_prop {
1307         text~ sans~ bold~ slanted
1308     }
1309 },
1310 BoldSlantedFeatures = {
1311     \prop_item:Nn \g__lngx_text_fonts_prop {
1312         text~ sans~ bold~ slanted~ features
1313     }
1314 },
1315 SwashFont = {

```

```

1316     \prop_item:Nn \g__lngx_text_fonts_prop {
1317         text~ sans~ swash
1318     }
1319 },
1320 SwashFeatures          = {
1321     \prop_item:Nn \g__lngx_text_fonts_prop {
1322         text~ sans~ swash~ features
1323     }
1324 },
1325 BoldSwashFont          = {
1326     \prop_item:Nn \g__lngx_text_fonts_prop {
1327         text~ sans~ bold~ swash
1328     }
1329 },
1330 BoldSwashFeatures      = {
1331     \prop_item:Nn \g__lngx_text_fonts_prop {
1332         text~ sans~ bold~ swash~ features
1333     }
1334 }
1335 } {
1336     \prop_item:Nn \g__lngx_text_fonts_prop {
1337         text~ sans~ upright
1338     }
1339 }
1340 \lngx_set_mono_font:ee {
1341     \prop_item:Nn \g__lngx_text_fonts_prop {
1342         text~ mono~ extra~ features
1343     },
1344     UprightFont          = {
1345         \prop_item:Nn \g__lngx_text_fonts_prop {
1346             text~ mono~ upright
1347         }
1348     },
1349     UprightFeatures      = {
1350         \prop_item:Nn \g__lngx_text_fonts_prop {
1351             text~ mono~ upright~ features
1352         }
1353     },
1354     ItalicFont           = {
1355         \prop_item:Nn \g__lngx_text_fonts_prop {
1356             text~ mono~ italic
1357         }
1358     },
1359     ItalicFeatures       = {
1360         \prop_item:Nn \g__lngx_text_fonts_prop {
1361             text~ mono~ italic~ features
1362         }
1363     },
1364     BoldFont             = {
1365         \prop_item:Nn \g__lngx_text_fonts_prop {
1366             text~ mono~ bold~ upright
1367         }
1368     },
1369     BoldFeatures         = {

```



```

I370     \prop_item:Nn \g__lngx_text_fonts_prop {
I371         text~ mono~ bold~ upright~ features
I372     }
I373 },
I374 BoldItalicFont      = {
I375     \prop_item:Nn \g__lngx_text_fonts_prop {
I376         text~ mono~ bold~ italic
I377     }
I378 },
I379 BoldItalicFeatures  = {
I380     \prop_item:Nn \g__lngx_text_fonts_prop {
I381         text~ mono~ bold~ italic~ features
I382     }
I383 },
I384 SlantedFont         = {
I385     \prop_item:Nn \g__lngx_text_fonts_prop {
I386         text~ mono~ slanted
I387     }
I388 },
I389 SlantedFeatures     = {
I390     \prop_item:Nn \g__lngx_text_fonts_prop {
I391         text~ mono~ slanted~ features
I392     }
I393 },
I394 BoldSlantedFont     = {
I395     \prop_item:Nn \g__lngx_text_fonts_prop {
I396         text~ mono~ bold~ slanted
I397     }
I398 },
I399 BoldSlantedFeatures = {
I400     \prop_item:Nn \g__lngx_text_fonts_prop {
I401         text~ mono~ bold~ slanted~ features
I402     }
I403 },
I404 SwashFont           = {
I405     \prop_item:Nn \g__lngx_text_fonts_prop {
I406         text~ mono~ swash
I407     }
I408 },
I409 SwashFeatures       = {
I410     \prop_item:Nn \g__lngx_text_fonts_prop {
I411         text~ mono~ swash~ features
I412     }
I413 },
I414 BoldSwashFont       = {
I415     \prop_item:Nn \g__lngx_text_fonts_prop {
I416         text~ mono~ bold~ swash
I417     }
I418 },
I419 BoldSwashFeatures   = {
I420     \prop_item:Nn \g__lngx_text_fonts_prop {
I421         text~ mono~ bold~ swash~ features
I422     }
I423 }

```

```

I424 } {
I425   \prop_item:Nn \g__lngx_text_fonts_prop {
I426     text~ mono~ upright
I427   }
I428 }
I429 \lngx_set_math_font:ee {
I430   \prop_item:Nn \g__lngx_text_fonts_prop {
I431     math~ features
I432   }
I433 } {
I434   \prop_item:Nn \g__lngx_text_fonts_prop { math }
I435 }
I436 }
I437 </font>

```

```

I438 \ipa
I439 \ProvidesExplPackage{linguistix-ipa}
I440 {2025-12-22}
I441 {v0.6a}
I442 {%
I443   A package for typesetting the IPA
I444   (International Phonetic Alphabet) from
I445   the ‘Linguistix’ bundle.%
I446 }

```

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

```

I447
I448 \IfPackageLoadedF { unicode-math } {
I449   \RequirePackage { unicode-math }
I450 }
I451
I452 \IfPackageLoadedF { linguistix-base } {
I453   \RequirePackage { linguistix-base }
I454 }
I455
I456 \IfPackageLoadedF { linguistix-nfss } {
I457   \RequirePackage { linguistix-nfss }
I458 }
I459
I460 \IfPackageLoadedF { linguistix-fixpex } {
I461   \RequirePackage { linguistix-fixpex }
I462 }

```

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

```

I463
I464 \NewDocumentCommand \ipatext { s m } {
I465   \IfBooleanTF { #1 } {
I466     {
I467       \lngxipa
I468       / #2 /
I469     }
I470   } {
I471     {
I472       \lngxipa
I473       [ #2 ]
I474     }
I475   }
I476 }

```

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

```

\g__lngx_ipa_fonts_prop
    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

```

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

All the keys here are prefixed with the word `ipa` in order to distinguish them from the keys provided by the `LINGUISCIX-FONTS` package. The argument of these keys should be expanded for which I use `\prop_gput:Nne` function. Each `#1` is replaced by the items from `clist` and the loop is repeated, whereas `##1` is the argument passed to the key by user.

```

\keys_define:nn { lngx_keys } {
  ipa~ #1
  .code:n = {
    \prop_gput:Nne \g__lngx_ipa_fonts_prop
      { ipa~ #1 }
      { ##1 }
  },
  ipa~ #1~ features
  .code:n = {
    \group_begin:
    \tl_clear:N \l_tmpa_tl
    \tl_clear:N \l_tmpb_tl
    \prop_get:Nn \g__lngx_ipa_fonts_prop
      { ipa~ #1~ features }
      \l_tmpa_tl
    \quark_if_no_value:NTF \l_tmpa_tl {
      \tl_set:Ne \l_tmpb_tl { ##1 }
    } {
      \tl_set:Ne \l_tmpb_tl { \l_tmpa_tl, ##1 }
    }
    \prop_gput:NnV \g__lngx_ipa_fonts_prop
      { ipa~ #1~ features }
      \l_tmpb_tl
    \group_end:
  }
}

```

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

`ipa extra features` This key adds to the property that stores the extra features for the serif fonts.

```

\keys_define:nn { lngx_keys } {

```

```

1520 ipa~ extra~ features
1521 .code:n          = {
1522   \group_begin:
1523   \tl_clear:N \l_tmpa_tl
1524   \tl_clear:N \l_tmpb_tl
1525   \prop_get:NeN \g__lngx_ipa_fonts_prop
1526               { ipa~ extra~ features }
1527               \l_tmpa_tl
1528   \quark_if_no_value:NTF \l_tmpa_tl {
1529     \tl_set:Ne \l_tmpb_tl { #1 }
1530   } {
1531     \tl_set:Ne \l_tmpb_tl { \l_tmpa_tl, #1 }
1532   }
1533   \prop_gput:NnV \g__lngx_ipa_fonts_prop
1534               { ipa~ extra~ features }
1535               \l_tmpb_tl
1536   \group_end:
1537 }
1538 }

```

(End of definition for ipa extra features. This function is documented on page 9.)

Since the only difference between the upcoming keys is that of the word **sans** and **mono**, we combine them together and use a nested **clist**. The rest of the mechanism is identical.

```

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
ipa mono upright
ipa mono upright features
ipa mono bold upright
ipa mono bold upright features
ipa mono italic
ipa mono italic features
ipa mono bold italic
ipa mono bold italic features
ipa mono slanted
ipa mono slanted features
ipa mono bold slanted
ipa mono bold slanted features
ipa mono swash
ipa mono swash features
ipa mono bold swash
ipa mono bold swash features
ipa mono small caps
ipa mono small caps features

```

```

1539
1540 \clist_map_inline:nn {
1541     sans,
1542     mono
1543 } {
1544     \clist_map_inline:nn {
1545         upright,
1546         bold~ upright,
1547         italic,
1548         bold~ italic,
1549         slanted,
1550         bold~ slanted,
1551         swash,
1552         bold~ swash,
1553         small~ caps
1554     } {
1555         \keys_define:nn { lngx_keys } {
1556             ipa~ #1~ ##1
1557             .code:n = {
1558                 \prop_gput:Nne \g__lngx_ipa_fonts_prop
1559                     { ipa~ #1~ ##1 }
1560                 { #####1 }
1561             },
1562             ipa~ #1~ ##1~ features
1563             .code:n = {
1564                 \group_begin:
1565                 \tl_clear:N \l_tmpa_tl
1566                 \tl_clear:N \l_tmpb_tl
1567                 \prop_get:Nen \g__lngx_ipa_fonts_prop
1568                     { ipa~ #1~ ##1~ features }
1569                     \l_tmpa_tl
1570                 \quark_if_no_value:NTF \l_tmpa_tl {
1571                     \tl_set:Nen \l_tmpb_tl { #####1 }
1572                 } {
1573                     \tl_set:Nen \l_tmpb_tl { \l_tmpa_tl, #####1 }
1574                 }
1575                 \prop_gput:NnV \g__lngx_ipa_fonts_prop
1576                     { ipa~ #1~ ##1~ features }
1577                     \l_tmpb_tl
1578                 \group_end:
1579             }
1580         }
1581     }
1582     \keys_define:nn { lngx_keys } {
1583         ipa~ #1~ extra~ features
1584         .code:n = {
1585             \group_begin:
1586             \tl_clear:N \l_tmpa_tl
1587             \tl_clear:N \l_tmpb_tl
1588             \prop_get:Nen \g__lngx_ipa_fonts_prop
1589                 { ipa~ #1~ extra~ features }
1590                 \l_tmpa_tl

```

```

1591     \quark_if_no_value:NTF \l_tmpa_tl {
1592       \tl_set:Ne \l_tmpb_tl { ##1 }
1593     } {
1594       \tl_set:Ne \l_tmpb_tl { \l_tmpa_tl, ##1 }
1595     }
1596     \prop_gput:NnV \g__lngx_ipa_fonts_prop
1597                   { ipa~ #1~ extra~ features }
1598                   \l_tmpb_tl
1599   \group_end:
1600 }
1601 }
1602 }

```

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

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

```

1603
1604 \keys_define:nn { lngx_keys } {
1605   ipa~ newcm
1606   .meta:n          = {
1607     ipa~ extra~
1608     features        = {
1609       IgnoreFontspecFile,
1610       StylisticSet   = {05}
1611     },
1612     ipa~ sans~ extra~
1613     features        = {
1614       IgnoreFontspecFile,
1615       StylisticSet   = {05}
1616     },
1617     ipa~ mono~ extra~
1618     features        = {
1619       IgnoreFontspecFile,
1620       StylisticSet   = {05}
1621     },
1622     ipa~
1623     upright          = {NewCM10-Book.otf},
1624     ipa~
1625     bold~ upright    = {NewCM10-Bold.otf},
1626     ipa~
1627     italic           = {NewCM10-BookItalic.otf},
1628     ipa~
1629     bold~ italic     = {NewCM10-BoldItalic.otf},
1630     ipa~
1631     slanted          = {NewCM10-Book.otf},
1632     ipa~
1633     bold~ slanted    = {NewCM10-Bold.otf},
1634     ipa~
1635     swash            = {NewCM10-Book.otf},
1636     ipa~
1637     bold~ swash      = {NewCM10-Bold.otf},
1638     ipa~ sans~

```

```

1639    upright                = {NewCMSans10-Book.otf},
1640    ipa~ sans~
1641    bold~ upright          = {NewCMSans10-Bold.otf},
1642    ipa~ sans~
1643    italic                 = {NewCMSans10-BookOblique.otf},
1644    ipa~ sans~
1645    bold~ italic           = {NewCMSans10-BoldOblique.otf},
1646    ipa~ sans~
1647    slanted                = {NewCMSans10-Book.otf},
1648    ipa~ sans~
1649    bold~ slanted          = {NewCMSans10-Bold.otf},
1650    ipa~ sans~
1651    swash                  = {NewCMSans10-Book.otf},
1652    ipa~ sans~
1653    bold~ swash            = {NewCMSans10-Bold.otf},
1654    ipa~ mono~
1655    upright                = {NewCMMono10-Book.otf},
1656    ipa~ mono~
1657    bold~ upright          = {NewCMMono10-Bold.otf},
1658    ipa~ mono~
1659    italic                 = {NewCMMono10-BookItalic.otf},
1660    ipa~ mono~
1661    bold~ italic           = {NewCMMono10-BoldOblique.otf},
1662    ipa~ mono~
1663    slanted                = {NewCMMono10-Book.otf},
1664    ipa~ mono~
1665    bold~ slanted          = {NewCMMono10-Bold.otf},
1666    ipa~ mono~
1667    swash                  = {NewCMMono10-Book.otf},
1668    ipa~ mono~
1669    bold~ swash            = {NewCMMono10-Bold.otf}
1670  }
1671 }

```

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

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

```

1672
1673 \keys_define:nn { lngx_keys } {
1674   ipa~ newcm~ sans
1675   .meta:n                = {
1676     ipa~
1677     upright                = {NewCMSans10-Book.otf},
1678     ipa~
1679     bold~ upright          = {NewCMSans10-Bold.otf},
1680     ipa~
1681     italic                 = {NewCMSans10-BookOblique.otf},
1682     ipa~
1683     bold~ italic           = {NewCMSans10-BoldOblique.otf},
1684     ipa~
1685     slanted                = {NewCMSans10-Book.otf},
1686     ipa~
1687     bold~ slanted          = {NewCMSans10-Bold.otf},
1688     ipa~

```



```

1689     swash                      = {NewCMSans10-Book.otf},
1690     ipa~                      = {NewCMSans10-Bold.otf}
1691     bold~ swash               = {NewCMSans10-Bold.otf}
1692   }
1693 }

```

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

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

```

1694
1695 \keys_define:nn { lngx_keys } {
1696   ipa~ newcm~ mono
1697   .meta:n                      = {
1698     ipa~
1699     upright                    = {NewCMMono10-Book.otf},
1700     ipa~
1701     bold~ upright              = {NewCMMono10-Bold.otf},
1702     ipa~
1703     italic                     = {NewCMMono10-BookItalic.otf},
1704     ipa~
1705     bold~ italic               = {NewCMMono10-BoldOblique.otf},
1706     ipa~
1707     slanted                    = {NewCMMono10-Book.otf},
1708     ipa~
1709     bold~ slanted              = {NewCMMono10-Bold.otf},
1710     ipa~
1711     swash                      = {NewCMMono10-Book.otf},
1712     ipa~
1713     bold~ swash                = {NewCMMono10-Bold.otf}
1714   }
1715 }

```

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

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

```

1716
1717 \keys_define:nn { lngx_keys } {
1718   ipa~ newcm~ regular
1719   .meta:n                      = {
1720     ipa~
1721     upright                    = {NewCM10-Regular.otf},
1722     ipa~
1723     bold~ upright              = {NewCM10-Bold.otf},
1724     ipa~
1725     italic                     = {NewCM10-Italic.otf},
1726     ipa~
1727     bold~ italic               = {NewCM10-BoldItalic.otf},
1728     ipa~
1729     slanted                    = {NewCM10-Regular.otf},
1730     ipa~
1731     bold~ slanted              = {NewCM10-Bold.otf},
1732     ipa~
1733     swash                      = {NewCM10-Regular.otf},

```

```

1734     ipa~
1735     bold~ swash                = {NewCM10-Bold.otf},
1736     ipa~ sans~
1737     upright                    = {NewCMSans10-Regular.otf},
1738     ipa~ sans~
1739     bold~ upright              = {NewCMSans10-Bold.otf},
1740     ipa~ sans~
1741     italic                     = {NewCMSans10-Oblique.otf},
1742     ipa~ sans~
1743     bold~ italic               = {NewCMSans10-BoldOblique.otf},
1744     ipa~ sans~
1745     slanted                    = {NewCMSans10-Regular.otf},
1746     ipa~ sans~
1747     bold~ slanted              = {NewCMSans10-Bold.otf},
1748     ipa~ sans~
1749     swash                      = {NewCMSans10-Regular.otf},
1750     ipa~ sans~
1751     bold~ swash                = {NewCMSans10-Bold.otf},
1752     ipa~ mono~
1753     upright                    = {NewCMMono10-Regular.otf},
1754     ipa~ mono~
1755     bold~ upright              = {NewCMMono10-Bold.otf},
1756     ipa~ mono~
1757     italic                     = {NewCMMono10-Italic.otf},
1758     ipa~ mono~
1759     bold~ italic               = {NewCMMono10-BoldOblique.otf},
1760     ipa~ mono~
1761     slanted                    = {NewCMMono10-Regular.otf},
1762     ipa~ mono~
1763     bold~ slanted              = {NewCMMono10-Bold.otf},
1764     ipa~ mono~
1765     swash                      = {NewCMMono10-Regular.otf},
1766     ipa~ mono~
1767     bold~ swash                = {NewCMMono10-Bold.otf}
1768   }
1769 }

```

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

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

```

1770
1771 \keys_define:nn { lngx_keys } {
1772   ipa~ newcm~ regular~ sans
1773   .meta:n                = {
1774     ipa
1775     upright                = {NewCMSans10-Regular.otf},
1776     ipa
1777     bold~ upright          = {NewCMSans10-Bold.otf},
1778     ipa
1779     italic                 = {NewCMSans10-Oblique.otf},
1780     ipa
1781     bold~ italic           = {NewCMSans10-BoldOblique.otf},
1782     ipa

```

```

1783     slanted                = {NewCMSans10-Regular.otf},
1784     ipa~
1785     bold~ slanted         = {NewCMSans10-Bold.otf},
1786     ipa~
1787     swash                 = {NewCMSans10-Regular.otf},
1788     ipa~
1789     bold~ swash           = {NewCMSans10-Bold.otf}
1790   }
1791 }

```

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

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

```

1792
1793 \keys_define:nn { lngx_keys } {
1794   ipa~ newcm~ regular~ mono
1795   .meta:n          = {
1796     ipa
1797     upright         = {NewCMMono10-Regular.otf},
1798     ipa
1799     bold~ upright   = {NewCMMono10-Bold.otf},
1800     ipa
1801     italic          = {NewCMMono10-Italic.otf},
1802     ipa
1803     bold~ italic    = {NewCMMono10-BoldOblique.otf},
1804     ipa~
1805     slanted         = {NewCMMono10-Regular.otf},
1806     ipa~
1807     bold~ slanted   = {NewCMMono10-Bold.otf},
1808     ipa~
1809     swash           = {NewCMMono10-Regular.otf},
1810     ipa~
1811     bold~ swash     = {NewCMMono10-Bold.otf}
1812   }
1813 }

```

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

We set the `ipa newcm` key by default.

```

1814
1815 \lngx_set_keys:n {ipa~ newcm}

```

If Lua^{La}TeX is loaded, the HarfBuzz renderer is selected by default.

```

1816
1817 \sys_if_engine luatex:T {
1818   \lngx_set_keys:n {
1819     ipa~ upright~
1820     features          = {
1821       Renderer        = { HarfBuzz }
1822     },
1823     ipa~ sans~ upright~
1824     features          = {
1825       Renderer        = { HarfBuzz }
1826     },
1827     ipa~ mono~ upright~

```

```

1828     features                = {
1829         Renderer              = { HarfBuzz }
1830     }
1831 }
1832 }

```

`\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.

```

1833 \lngx_set_main_ipa_font:nn
1834 \lngx_main_ipa:
1835 \lngx_ipa_rm_nfss
1836 \lngx_set_sans_ipa_font:nn
1837 \lngx_sans_ipa:
1838 \lngx_ipa_sf_nfss
1839 \lngx_set_mono_ipa_font:nn
1840 \lngx_mono_ipa:
1841 \lngx_ipa_tt_nfss
1842
1843 \cs_new_protected:Npn \lngx_set_main_ipa_font:nn #1#2 {
1844     \setfontfamily \lngx_main_ipa: [
1845         #1,
1846         NFSSFamily              = { lngx_ipa_rm_nfss }
1847     ] { #2 }
1848 }
1849
1850 \cs_new_protected:Npn \lngx_set_sans_ipa_font:nn #1#2 {
1851     \setfontfamily \lngx_sans_ipa: [
1852         #1,
1853         NFSSFamily              = { lngx_ipa_sf_nfss }
1854     ] { #2 }
1855 }
1856
1857 \cs_new_protected:Npn \lngx_set_mono_ipa_font:nn #1#2 {
1858     \setfontfamily \lngx_mono_ipa: [
1859         #1,
1860         NFSSFamily              = { lngx_ipa_tt_nfss }
1861     ] { #2 }
1862 }
1863
1864 \cs_generate_variant:Nn \lngx_set_main_ipa_font:nn { ee }
1865 \cs_generate_variant:Nn \lngx_set_sans_ipa_font:nn { ee }
1866 \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 15.)

`lngx_ipa` Here, I create a ‘super font family’ with `\lngx_super_font_family:nn`, a macro provided by LINGUIS~~TeX~~_{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.

```

1858
1859 \lngx_super_font_family:nn { lngx_ipa } {
1860     rm              = { lngx_ipa_rm_nfss },
1861     sf              = { lngx_ipa_sf_nfss },
1862     tt              = { lngx_ipa_tt_nfss }
1863 }

```

(End of definition for `lngx_ipa`. This function is documented on page 16.)

`\lngxipa` I use `\lngx softer super font family:n` provided by LINGUIS~~TeX~~_{NFSS} for defining this switch to the IPA.

```

1864
1865 \cs_new_protected:Npn \lngx_ipa: {
1866   \lngx_softer_super_font_family:n { lngx_ipa }
1867 }
1868
1869 \cs_gset_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 LINGUIS~~T~~**X**-FONTS for setting the size variants. This is done with lazy evaluation, just before `\begin{document}`.

```

1870
1871 \hook_gput_code:nnn { begindocument / before } { . } {
1872   \group_begin:
1873   \clist_map_inline:nn {
1874     upright,
1875     slanted,
1876     swash
1877   } {
1878     \prop_get:NnN \g__lngx_ipa_fonts_prop
1879                 { ipa~ #1 }
1880                 \l_tmpa_tl
1881     \tl_if_eq:NnT \l_tmpa_tl { NewCM10-Book.otf } {
1882       \lngx_set_keys:n {
1883         ipa~ #1~
1884         features = {
1885           SizeFeatures = {
1886             {
1887               Size = {-8},
1888               Font = {
1889                 NewCM08-Book.otf
1890             }
1891           },
1892           {
1893             Size = {8-},
1894             Font = {
1895               NewCM10-Book.otf
1896             }
1897         }
1898       }
1899     }
1900   }
1901 }
1902 \tl_if_eq:NnT \l_tmpa_tl { NewCM10-Regular.otf } {
1903   \lngx_set_keys:n {
1904     ipa~ #1~
1905     features = {
1906       SizeFeatures = {
1907         {
1908           Size = {-8},
1909           Font = {
1910             NewCM08-Regular.otf
1911         }
1912       },

```

```

1913         {
1914             Size                = {8-},
1915             Font                = {
1916                 NewCM10-Regular.otf
1917             }
1918         }
1919     }
1920 }
1921 }
1922 }
1923 \tl_if_eq:NnT \l_tmpa_tl { NewCMSans10-Book.otf } {
1924     \lngx_set_keys:n {
1925         ipa~ #1~
1926         features                = {
1927             SizeFeatures        = {
1928                 {
1929                     Size                = {-8},
1930                     Font                = {
1931                         NewCMSans08-Book.otf
1932                     }
1933                 },
1934                 {
1935                     Size                = {8-},
1936                     Font                = {
1937                         NewCMSans10-Book.otf
1938                     }
1939                 }
1940             }
1941         }
1942     }
1943 }
1944 \tl_if_eq:NnT \l_tmpa_tl { NewCMSans10-Regular.otf } {
1945     \lngx_set_keys:n {
1946         ipa~ #1~
1947         features                = {
1948             SizeFeatures        = {
1949                 {
1950                     Size                = {-8},
1951                     Font                = {
1952                         NewCMSans08-Regular.otf
1953                     }
1954                 },
1955                 {
1956                     Size                = {8-},
1957                     Font                = {
1958                         NewCMSans10-Regular.otf
1959                     }
1960                 }
1961             }
1962         }
1963     }
1964 }
1965 \tl_if_eq:NnT \l_tmpa_tl { NewCMMono10-Book.otf } {
1966     \lngx_set_keys:n {

```

```

1967     ipa~ #1~
1968     features                                = {
1969         SizeFeatures                        = {
1970             {
1971                 Size                        = {0-},
1972                 Font                        = {
1973                     NewCMMono10-Book.otf
1974                 }
1975             }
1976         }
1977     }
1978 }
1979
1980 \tl_if_eq:NnT \l_tmpa_tl { NewCMMono10-Regular.otf } {
1981     \lngx_set_keys:n {
1982         ipa~ #1~
1983         features                                = {
1984             SizeFeatures                        = {
1985                 {
1986                     Size                        = {0-},
1987                     Font                        = {
1988                         NewCMMono10-Regular.otf
1989                     }
1990                 }
1991             }
1992         }
1993     }
1994 }
1995 \prop_get:NnN \g__lngx_ipa_fonts_prop
1996     { ipa~ sans~ #1 }
1997     \l_tmpa_tl
1998 \tl_if_eq:NnT \l_tmpa_tl { NewCMSans10-Book.otf } {
1999     \lngx_set_keys:n {
2000         ipa~ sans~ #1~
2001         features                                = {
2002             SizeFeatures                        = {
2003                 {
2004                     Size                        = {-8},
2005                     Font                        = {
2006                         NewCMSans08-Book.otf
2007                     }
2008                 },
2009                 {
2010                     Size                        = {8-},
2011                     Font                        = {
2012                         NewCMSans10-Book.otf
2013                     }
2014                 }
2015             }
2016         }
2017     }
2018 }
2019 \tl_if_eq:NnT \l_tmpa_tl { NewCMSans10-Regular.otf } {
2020     \lngx_set_keys:n {

```

```

2021     ipa~ sans~ #1~
2022     features                                = {
2023         SizeFeatures                        = {
2024             {
2025                 Size                        = {-8},
2026                 Font                        = {
2027                     NewCMSans08-Regular.otf
2028                 }
2029             },
2030             {
2031                 Size                        = {8-},
2032                 Font                        = {
2033                     NewCMSans10-Regular.otf
2034                 }
2035             }
2036         }
2037     }
2038 }
2039 }
2040 \prop_get:NnN \g__lngx_ipa_fonts_prop
2041     { ipa~ mono~ #1 }
2042     \l_tmpa_tl
2043 \tl_if_eq:NnT \l_tmpa_tl { NewCMMono10-Book.otf } {
2044     \lngx_set_keys:n {
2045         ipa~ mono~ #1~
2046         features                                = {
2047             SizeFeatures                        = {
2048                 {
2049                     Size                        = {0-},
2050                     Font                        = {
2051                         NewCMMono10-Book.otf
2052                     }
2053                 }
2054             }
2055         }
2056     }
2057 }
2058 \tl_if_eq:NnT \l_tmpa_tl { NewCMMono10-Regular.otf } {
2059     \lngx_set_keys:n {
2060         ipa~ mono~ #1~
2061         features                                = {
2062             SizeFeatures                        = {
2063                 {
2064                     Size                        = {0-},
2065                     Font                        = {
2066                         NewCMMono10-Regular.otf
2067                     }
2068                 }
2069             }
2070         }
2071     }
2072 }
2073 }
2074 \clist_map_inline:nn {

```



```

2075     upright,
2076     slanted,
2077     swash
2078 } {
2079   \prop_get:NnN \g__lngx_ipa_fonts_prop
2080   { ipa~ bold~ #1 }
2081   \l_tmpa_tl
2082   \tl_if_eq:NnT \l_tmpa_tl { NewCM10-Bold.otf } {
2083     \lngx_set_keys:n {
2084       ipa~ bold~ #1~
2085       features          = {
2086         SizeFeatures    = {
2087           {
2088             Size        = {0-},
2089             Font        = {
2090               NewCM10-Bold.otf
2091             }
2092           }
2093         }
2094       }
2095     }
2096   }
2097   \tl_if_eq:NnT \l_tmpa_tl { NewCMSans10-Bold.otf } {
2098     \lngx_set_keys:n {
2099       ipa~ bold~ #1~
2100       features          = {
2101         {
2102           Size          = {0-},
2103           Font          = {
2104             NewCMSans10-Bold.otf
2105           }
2106         }
2107       }
2108     }
2109   }
2110   \tl_if_eq:NnT \l_tmpa_tl { NewCMMono10-Bold.otf } {
2111     \lngx_set_keys:n {
2112       ipa~ bold~ #1~
2113       features          = {
2114         SizeFeatures    = {
2115           {
2116             Size        = {0-},
2117             Font        = {
2118               NewCMMono10-Bold.otf
2119             }
2120           }
2121         }
2122       }
2123     }
2124   }
2125   \prop_get:NnN \g__lngx_ipa_fonts_prop
2126   { ipa~ sans~ bold~ #1 }
2127   \l_tmpa_tl
2128   \tl_if_eq:NnT \l_tmpa_tl { NewCMSans10-Bold.otf } {

```

```

2129 \ltx_set_keys:n {
2130   ipa~ sans~ bold~ #1~
2131   features = {
2132     SizeFeatures = {
2133       {
2134         Size = {0-},
2135         Font = {
2136           NewCMSans10-Bold.otf
2137         }
2138       }
2139     }
2140   }
2141 }
2142
2143 \prop_get:NnN \g__ltx_ipa_fonts_prop
2144 { ipa~ mono~ bold~ #1 }
2145 \l_tmpa_tl
2146 \tl_if_eq:NnT \l_tmpa_tl { NewCMMono10-Bold.otf } {
2147   \ltx_set_keys:n {
2148     ipa~ mono~ bold~ #1~
2149     features = {
2150       SizeFeatures = {
2151         {
2152           Size = {0-},
2153           Font = {
2154             NewCMMono10-Bold.otf
2155           }
2156         }
2157       }
2158     }
2159   }
2160 }
2161 }
2162 \prop_get:NnN \g__ltx_ipa_fonts_prop
2163 { ipa~ italic }
2164 \l_tmpa_tl
2165 \tl_if_eq:NnT \l_tmpa_tl { NewCM10-BookItalic.otf } {
2166   \ltx_set_keys:n {
2167     ipa~ italic~
2168     features = {
2169       SizeFeatures = {
2170         {
2171           Size = {-8},
2172           Font = {
2173             NewCM08-BookItalic.otf
2174           }
2175         },
2176         {
2177           Size = {8-},
2178           Font = {
2179             NewCM10-BookItalic.otf
2180           }
2181         }
2182       }

```

```

2183     }
2184   }
2185 }
2186 \tl_if_eq:NnT \l_tmpa_tl { NewCM10-Italic.otf } {
2187   \lngx_set_keys:n {
2188     ipa~ italic~
2189     features = {
2190       SizeFeatures = {
2191         {
2192           Size = {-8},
2193           Font = {
2194             NewCM08-Italic.otf
2195           }
2196         },
2197         {
2198           Size = {8-},
2199           Font = {
2200             NewCM10-Italic.otf
2201           }
2202         }
2203       }
2204     }
2205   }
2206 }
2207 \tl_if_eq:NnT \l_tmpa_tl { NewCMSans10-BookOblique.otf } {
2208   \lngx_set_keys:n {
2209     ipa~ italic~
2210     features = {
2211       SizeFeatures = {
2212         {
2213           Size = {-8},
2214           Font = {
2215             NewCMSans08-BookOblique.otf
2216           }
2217         },
2218         {
2219           Size = {8-},
2220           Font = {
2221             NewCMSans10-BookOblique.otf
2222           }
2223         }
2224       }
2225     }
2226   }
2227 }
2228 \tl_if_eq:NnT \l_tmpa_tl { NewCMSans10-Oblique.otf } {
2229   \lngx_set_keys:n {
2230     ipa~ italic~
2231     features = {
2232       SizeFeatures = {
2233         {
2234           Size = {-8},
2235           Font = {
2236             NewCMSans08-Oblique.otf

```

```

2237     }
2238   },
2239   {
2240     Size              = {8-},
2241     Font              = {
2242       NewCMSans10-Oblique.otf
2243     }
2244   }
2245 }
2246 }
2247 }
2248 }
2249 \tl_if_eq:NnT \l_tmpa_tl { NewCMMono10-BookItalic.otf } {
2250   \lngx_set_keys:n {
2251     ipa~ italic~
2252     features          = {
2253       SizeFeatures    = {
2254         {
2255           Size        = {0-},
2256           Font        = {
2257             NewCMMono10-BookItalic.otf
2258           }
2259         }
2260       }
2261     }
2262   }
2263 }
2264 \tl_if_eq:NnT \l_tmpa_tl { NewCMMono10-Italic.otf } {
2265   \lngx_set_keys:n {
2266     ipa~ italic~
2267     features          = {
2268       SizeFeatures    = {
2269         {
2270           Size        = {0-},
2271           Font        = {
2272             NewCMMono10-Italic.otf
2273           }
2274         }
2275       }
2276     }
2277   }
2278 }
2279 \prop_get:NnN \g__lngx_ipa_fonts_prop
2280   { ipa~ sans~ italic }
2281   \l_tmpa_tl
2282 \tl_if_eq:NnT \l_tmpa_tl { NewCMSans10-BookOblique.otf } {
2283   \lngx_set_keys:n {
2284     ipa~ sans~ italic~
2285     features          = {
2286       SizeFeatures    = {
2287         {
2288           Size        = {-8},
2289           Font        = {
2290             NewCMSans08-BookOblique.otf

```

```

2291     }
2292   },
2293   {
2294     Size              = {8-},
2295     Font              = {
2296       NewCMSans10-BookOblique.otf
2297     }
2298   }
2299 }
2300 }
2301 }
2302 }
2303 \tl_if_eq:NnT \l_tmpa_tl { NewCMSans10-Oblique.otf } {
2304   \lngx_set_keys:n {
2305     ipa~ sans~ italic~
2306     features          = {
2307       SizeFeatures    = {
2308         {
2309           Size        = {-8},
2310           Font        = {
2311             NewCMSans08-Oblique.otf
2312           }
2313         },
2314         {
2315           Size        = {8-},
2316           Font        = {
2317             NewCMSans10-Oblique.otf
2318           }
2319         }
2320       }
2321     }
2322   }
2323 }
2324 \prop_get:NnN \g__lngx_ipa_fonts_prop
2325   { ipa~ mono~ italic }
2326   \l_tmpa_tl
2327 \tl_if_eq:NnT \l_tmpa_tl { NewCMMono10-BookItalic.otf } {
2328   \lngx_set_keys:n {
2329     ipa~ mono~ italic~
2330     features          = {
2331       SizeFeatures    = {
2332         {
2333           Size        = {0-},
2334           Font        = {
2335             NewCMMono10-BookItalic.otf
2336           }
2337         },
2338         {
2339           Size        = {8-},
2340           Font        = {
2341             NewCMMono10-Italic.otf
2342           }
2343         }
2344       }
2345     }
2346   }
2347 }
2348 \tl_if_eq:NnT \l_tmpa_tl { NewCMMono10-Italic.otf } {
2349   \lngx_set_keys:n {
2350     ipa~ mono~ italic~

```

```

2345     features                                = {
2346         SizeFeatures                        = {
2347             {
2348                 Size                        = {0-},
2349                 Font                        = {
2350                     NewCMMono10-Italic.otf
2351                 }
2352             }
2353         }
2354     }
2355 }
2356 }
2357 \prop_get:NnN \g__lngx_ipa_fonts_prop
2358     { ipa~ bold~ italic }
2359     \l_tmpa_tl
2360 \tl_if_eq:NnT \l_tmpa_tl { NewCM10-BoldItalic.otf } {
2361     \lngx_set_keys:n {
2362         ipa~ bold~ italic~
2363         features                                = {
2364             SizeFeatures                        = {
2365                 {
2366                     Size                        = {0-},
2367                     Font                        = {
2368                         NewCM10-BoldItalic.otf
2369                     }
2370                 }
2371             }
2372         }
2373     }
2374 }
2375 \tl_if_eq:NnT \l_tmpa_tl { NewCMSans10-BoldOblique.otf } {
2376     \lngx_set_keys:n {
2377         ipa~ sans~ bold~ italic~
2378         features                                = {
2379             SizeFeatures                        = {
2380                 {
2381                     Size                        = {0-},
2382                     Font                        = {
2383                         NewCMSans10-BoldOblique.otf
2384                     }
2385                 }
2386             }
2387         }
2388     }
2389 }
2390 \tl_if_eq:NnT \l_tmpa_tl { NewCMMono10-BoldOblique.otf } {
2391     \lngx_set_keys:n {
2392         ipa~ mono~ bold~ italic~
2393         features                                = {
2394             SizeFeatures                        = {
2395                 {
2396                     Size                        = {0-},
2397                     Font                        = {
2398                         NewCMMono10-BoldOblique.otf

```

```

2399     }
2400   }
2401 }
2402 }
2403 }
2404 }
2405 \prop_get:NnN \g__lngx_ipa_fonts_prop
2406   { ipa~ sans~ bold~ italic }
2407   \l_tmpa_tl
2408 \tl_if_eq:NnT \l_tmpa_tl { NewCMSans10-BoldOblique.otf } {
2409   \lngx_set_keys:n {
2410     ipa~ sans~ bold~ italic~
2411     features = {
2412       SizeFeatures = {
2413         {
2414           Size = {0-},
2415           Font = {
2416             NewCMSans10-BoldOblique.otf
2417           }
2418         }
2419       }
2420     }
2421   }
2422 }
2423 \prop_get:NnN \g__lngx_ipa_fonts_prop
2424   { ipa~ mono~ bold~ italic }
2425   \l_tmpa_tl
2426 \tl_if_eq:NnT \l_tmpa_tl { NewCMMono10-BoldOblique.otf } {
2427   \lngx_set_keys:n {
2428     ipa~ mono~ bold~ italic~
2429     features = {
2430       SizeFeatures = {
2431         {
2432           Size = {0-},
2433           Font = {
2434             NewCMMono10-BoldOblique.otf
2435           }
2436         }
2437       }
2438     }
2439   }
2440 }
2441 \group_end:
2442 \lngx_set_main_ipa_font:ee {
2443   \prop_item:Nn \g__lngx_ipa_fonts_prop {
2444     ipa~ extra~ features
2445   },
2446   UprightFont = {
2447     \prop_item:Nn \g__lngx_ipa_fonts_prop {
2448       ipa~ upright
2449     }
2450   },
2451   UprightFeatures = {
2452     \prop_item:Nn \g__lngx_ipa_fonts_prop {

```

```

2453     ipa~ upright~ features
2454 }
2455 },
2456 ItalicFont = {
2457     \prop_item:Nn \g__lngx_ipa_fonts_prop {
2458         ipa~ italic
2459     }
2460 },
2461 ItalicFeatures = {
2462     \prop_item:Nn \g__lngx_ipa_fonts_prop {
2463         ipa~ italic~ features
2464     }
2465 },
2466 BoldFont = {
2467     \prop_item:Nn \g__lngx_ipa_fonts_prop {
2468         ipa~ bold~ upright
2469     }
2470 },
2471 BoldFeatures = {
2472     \prop_item:Nn \g__lngx_ipa_fonts_prop {
2473         ipa~ bold~ upright~ features
2474     }
2475 },
2476 BoldItalicFont = {
2477     \prop_item:Nn \g__lngx_ipa_fonts_prop {
2478         ipa~ bold~ italic
2479     }
2480 },
2481 BoldItalicFeatures = {
2482     \prop_item:Nn \g__lngx_ipa_fonts_prop {
2483         ipa~ bold~ italic~ features
2484     }
2485 },
2486 SlantedFont = {
2487     \prop_item:Nn \g__lngx_ipa_fonts_prop {
2488         ipa~ slanted
2489     }
2490 },
2491 SlantedFeatures = {
2492     \prop_item:Nn \g__lngx_ipa_fonts_prop {
2493         ipa~ slanted~ features
2494     }
2495 },
2496 BoldSlantedFont = {
2497     \prop_item:Nn \g__lngx_ipa_fonts_prop {
2498         ipa~ bold~ slanted
2499     }
2500 },
2501 BoldSlantedFeatures = {
2502     \prop_item:Nn \g__lngx_ipa_fonts_prop {
2503         ipa~ bold~ slanted~ features
2504     }
2505 },
2506 SwashFont = {

```



```

2507     \prop_item:Nn \g__lngx_ipa_fonts_prop {
2508       ipa~ swash
2509     }
2510   },
2511   SwashFeatures = {
2512     \prop_item:Nn \g__lngx_ipa_fonts_prop {
2513       ipa~ swash~ features
2514     }
2515   },
2516   BoldSwashFont = {
2517     \prop_item:Nn \g__lngx_ipa_fonts_prop {
2518       ipa~ bold~ swash
2519     }
2520   },
2521   BoldSwashFeatures = {
2522     \prop_item:Nn \g__lngx_ipa_fonts_prop {
2523       ipa~ bold~ swash~ features
2524     }
2525   }
2526 } {
2527   \prop_item:Nn \g__lngx_ipa_fonts_prop {
2528     ipa~ upright
2529   }
2530 }
2531 \lngx_set_sans_ipa_font:ee {
2532   \prop_item:Nn \g__lngx_ipa_fonts_prop {
2533     ipa~ sans~ extra~ features
2534   },
2535   UprightFont = {
2536     \prop_item:Nn \g__lngx_ipa_fonts_prop {
2537       ipa~ sans~ upright
2538     }
2539   },
2540   UprightFeatures = {
2541     \prop_item:Nn \g__lngx_ipa_fonts_prop {
2542       ipa~ sans~ upright~ features
2543     }
2544   },
2545   ItalicFont = {
2546     \prop_item:Nn \g__lngx_ipa_fonts_prop {
2547       ipa~ sans~ italic
2548     }
2549   },
2550   ItalicFeatures = {
2551     \prop_item:Nn \g__lngx_ipa_fonts_prop {
2552       ipa~ sans~ italic~ features
2553     }
2554   },
2555   BoldFont = {
2556     \prop_item:Nn \g__lngx_ipa_fonts_prop {
2557       ipa~ sans~ bold~ upright
2558     }
2559   },
2560   BoldFeatures = {

```

```

2561     \prop_item:Nn \g__lngx_ipa_fonts_prop {
2562       ipa~ sans~ bold~ upright~ features
2563     }
2564   },
2565   BoldItalicFont      = {
2566     \prop_item:Nn \g__lngx_ipa_fonts_prop {
2567       ipa~ sans~ bold~ italic
2568     }
2569   },
2570   BoldItalicFeatures  = {
2571     \prop_item:Nn \g__lngx_ipa_fonts_prop {
2572       ipa~ sans~ bold~ italic~ features
2573     }
2574   },
2575   SlantedFont         = {
2576     \prop_item:Nn \g__lngx_ipa_fonts_prop {
2577       ipa~ sans~ slanted
2578     }
2579   },
2580   SlantedFeatures     = {
2581     \prop_item:Nn \g__lngx_ipa_fonts_prop {
2582       ipa~ sans~ slanted~ features
2583     }
2584   },
2585   BoldSlantedFont     = {
2586     \prop_item:Nn \g__lngx_ipa_fonts_prop {
2587       ipa~ sans~ bold~ slanted
2588     }
2589   },
2590   BoldSlantedFeatures = {
2591     \prop_item:Nn \g__lngx_ipa_fonts_prop {
2592       ipa~ sans~ bold~ slanted~ features
2593     }
2594   },
2595   SwashFont           = {
2596     \prop_item:Nn \g__lngx_ipa_fonts_prop {
2597       ipa~ sans~ swash
2598     }
2599   },
2600   SwashFeatures       = {
2601     \prop_item:Nn \g__lngx_ipa_fonts_prop {
2602       ipa~ sans~ swash~ features
2603     }
2604   },
2605   BoldSwashFont       = {
2606     \prop_item:Nn \g__lngx_ipa_fonts_prop {
2607       ipa~ sans~ bold~ swash
2608     }
2609   },
2610   BoldSwashFeatures   = {
2611     \prop_item:Nn \g__lngx_ipa_fonts_prop {
2612       ipa~ sans~ bold~ swash~ features
2613     }
2614   }

```

```

2615 } {
2616   \prop_item:Nn \g__lngx_ipa_fonts_prop {
2617     ipa~ sans~ upright
2618   }
2619 }
2620 \lngx_set_mono_ipa_font:ee {
2621   \prop_item:Nn \g__lngx_ipa_fonts_prop {
2622     ipa~ mono~ extra~ features
2623   },
2624   UprightFont = {
2625     \prop_item:Nn \g__lngx_ipa_fonts_prop {
2626       ipa~ mono~ upright
2627     }
2628   },
2629   UprightFeatures = {
2630     \prop_item:Nn \g__lngx_ipa_fonts_prop {
2631       ipa~ mono~ upright~ features
2632     }
2633   },
2634   ItalicFont = {
2635     \prop_item:Nn \g__lngx_ipa_fonts_prop {
2636       ipa~ mono~ italic
2637     }
2638   },
2639   ItalicFeatures = {
2640     \prop_item:Nn \g__lngx_ipa_fonts_prop {
2641       ipa~ mono~ italic~ features
2642     }
2643   },
2644   BoldFont = {
2645     \prop_item:Nn \g__lngx_ipa_fonts_prop {
2646       ipa~ mono~ bold~ upright
2647     }
2648   },
2649   BoldFeatures = {
2650     \prop_item:Nn \g__lngx_ipa_fonts_prop {
2651       ipa~ mono~ bold~ upright~ features
2652     }
2653   },
2654   BoldItalicFont = {
2655     \prop_item:Nn \g__lngx_ipa_fonts_prop {
2656       ipa~ mono~ bold~ italic
2657     }
2658   },
2659   BoldItalicFeatures = {
2660     \prop_item:Nn \g__lngx_ipa_fonts_prop {
2661       ipa~ mono~ bold~ italic~ features
2662     }
2663   },
2664   SlantedFont = {
2665     \prop_item:Nn \g__lngx_ipa_fonts_prop {
2666       ipa~ mono~ slanted
2667     }
2668   },

```

```

2669 SlantedFeatures          = {
2670     \prop_item:Nn \g__lngx_ipa_fonts_prop {
2671         ipa~ mono~ slanted~ features
2672     }
2673 },
2674 BoldSlantedFont          = {
2675     \prop_item:Nn \g__lngx_ipa_fonts_prop {
2676         ipa~ mono~ bold~ slanted
2677     }
2678 },
2679 BoldSlantedFeatures      = {
2680     \prop_item:Nn \g__lngx_ipa_fonts_prop {
2681         ipa~ mono~ bold~ slanted~ features
2682     }
2683 },
2684 SwashFont                = {
2685     \prop_item:Nn \g__lngx_ipa_fonts_prop {
2686         ipa~ mono~ swash
2687     }
2688 },
2689 SwashFeatures            = {
2690     \prop_item:Nn \g__lngx_ipa_fonts_prop {
2691         ipa~ mono~ swash~ features
2692     }
2693 },
2694 BoldSwashFont            = {
2695     \prop_item:Nn \g__lngx_ipa_fonts_prop {
2696         ipa~ mono~ bold~ swash
2697     }
2698 },
2699 BoldSwashFeatures        = {
2700     \prop_item:Nn \g__lngx_ipa_fonts_prop {
2701         ipa~ mono~ bold~ swash~ features
2702     }
2703 }
2704 } {
2705     \prop_item:Nn \g__lngx_ipa_fonts_prop {
2706         ipa~ mono~ upright
2707     }
2708 }
2709 }
2710 </ipa>

```

```

2711 (*lang)
2712 \ProvidesExplPackage{linguistix-languages}
2713     {2025-05-21}
2714     {v0.1}
2715     {%
2716         An assistant package for automatically
2717         loading fonts and more settings for
2718         languages.%
2719     }

```

LINGUISTIX-BASE is loaded (if not already done) for the key-value parser.

```

2720
2721 \IfPackageLoadedF { linguistix-base } {
2722     \RequirePackage { linguistix-base }
2723 }

```

The `babel` package is loaded with `provide**` option as it mandates the use of modern mechanism.

```

2724
2725 \IfPackageLoadedF { babel } {
2726     \RequirePackage [ provide * = * ] { babel }
2727 }

```

`\g_lngx_main_language_tl` I declare a `tl` that I will use for storing the main language. It is publicly available.

```

2728
2729 \tl_new:N \g_lngx_main_language_tl

```

(End of definition for `\g_lngx_main_language_tl`. This function is documented on page 16.)

`\g_lngx_languages_clist` I declare a `clist` that I will use for storing languages. It is publicly available.

```

2730
2731 \clist_new:N \g_lngx_languages_clist

```

(End of definition for `\g_lngx_languages_clist`. This function is documented on page 16.)

`\lngx_languages:nn` I develop a wrapper macro with a `:VV` variant.

`\providelanguage`

```

2732
2733 \cs_new_protected:Npn \lngx_languages:nn #1#2 {
2734     \babelprovide [ #1 ] { #2 }
2735 }
2736
2737 \cs_generate_variant:Nn \lngx_languages:nn { VV }
2738 \cs_gset_eq:NN \providelanguage \lngx_languages:nn

```

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

The `babel` package produces an ‘info’ message if the fonts are not set with `\babelfont`. Mostly they aren’t set with this mechanism, so this warning is inevitable in default situations. Imagine loading LINGUISTIX-FONTS first and then loading this package. The fonts are already set with `\setmainfont` and friends. Thus we will be prompted with this warning always. In order to avoid that, I renew the wrapper functions around `\setmainfont` to `\babelfont`. Note that this only affects the usage when LINGUISTIX-FONTS is loaded. If you use LINGUISTIX-LANGUAGES and then use `\setmainfont`-like commands, you will get `babel`’s warning and I have no intention to suppress that behaviour.

```

2739
2740 \IfPackageLoadedTF { linguistix-fonts } {
2741   \cs_gset_protected:Npn \lngx_set_main_font:nn #1#2 {
2742     \babelfont { rm } [ #1 ] { #2 }
2743   }
2744   \cs_gset_protected:Npn \lngx_set_sans_font:nn #1#2 {
2745     \babelfont { sf } [ #1 ] { #2 }
2746   }
2747   \cs_gset_protected:Npn \lngx_set_mono_font:nn #1#2 {
2748     \babelfont { tt } [ #1 ] { #2 }
2749   }
2750 } {
2751   \cs_new_protected:Npn \lngx_set_main_font:nn #1#2 {
2752     \babelfont { rm } [ #1 ] { #2 }
2753   }
2754   \cs_new_protected:Npn \lngx_set_sans_font:nn #1#2 {
2755     \babelfont { sf } [ #1 ] { #2 }
2756   }
2757   \cs_new_protected:Npn \lngx_set_mono_font:nn #1#2 {
2758     \babelfont { tt } [ #1 ] { #2 }
2759   }
2760 }

```

`\lngx_other_main_font:nnn` The following macros set fonts for other languages using the `\babelfont` command.

`\lngx_other_sans_font:nnn`

`\lngx_other_mono_font:nnn`

```

2761
2762 \cs_gset_protected:Npn \lngx_other_main_font:nnn #1#2#3 {
2763   \babelfont [ #1 ] { rm } [ #2 ] { #3 }
2764 }
2765
2766 \cs_gset_protected:Npn \lngx_other_sans_font:nnn #1#2#3 {
2767   \babelfont [ #1 ] { sf } [ #2 ] { #3 }
2768 }
2769
2770 \cs_gset_protected:Npn \lngx_other_mono_font:nnn #1#2#3 {
2771   \babelfont [ #1 ] { tt } [ #2 ] { #3 }
2772 }
2773
2774 \cs_generate_variant:Nn \lngx_other_main_font:nnn { nee }
2775 \cs_generate_variant:Nn \lngx_other_sans_font:nnn { nee }
2776 \cs_generate_variant:Nn \lngx_other_mono_font:nnn { nee }

```

(End of definition for `\lngx_other_main_font:nnn`, `\lngx_other_sans_font:nnn`, and `\lngx_other_mono_font:nnn`. These functions are documented on page 15.)

`\lngx_load_languages:n` I provide a simple macro that only does the job of loading languages, both in L^AT_EX₃
`\loadlanguages` style, as well as the in the plain style.

```

2777
2778 \cs_new_protected:Npn \lngx_load_languages:n #1 {
2779   \lngx_set_keys:n { languages = { #1 } }
2780 }
2781
2782 \cs_gset_eq:NN \loadlanguages \lngx_load_languages:n

```

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

I equate the `\arabic` command to a new command I want to provide. This is done in order to get control over the default L^AT_EX counters. The command is manipulated when plugs are activated.

`\lngx_counter:n`

```
2783
2784 \cs_gset_eq:NN \lngx_counter:n \arabic
```

(End of definition for `\lngx_counter:n`. This function is documented on page 16.)

Now all the default counters are changed from `\arabic` to `\lngx_counter:n`.

```
2785
2786 \cs_set:Npn \thechapter {
2787   \lngx_counter:n { chapter }
2788 }
2789 \cs_set:Npn \thesection {
2790   \lngx_counter:n { section }
2791 }
2792 \cs_set:Npn \thesubsection {
2793   \lngx_counter:n { subsection }
2794 }
2795 \cs_set:Npn \thesubsubsection {
2796   \lngx_counter:n { subsubsection }
2797 }
2798 \cs_set:Npn \theparagraph {
2799   \lngx_counter:n { section }
2800 }
2801 \cs_set:Npn \thesubparagraph {
2802   \lngx_counter:n { section }
2803 }
2804 \cs_set:Npn \thepage {
2805   \lngx_counter:n { page }
2806 }
2807 \cs_set:Npn \thefigure {
2808   \lngx_counter:n { figure }
2809 }
2810 \cs_set:Npn \thetable {
2811   \lngx_counter:n { table }
2812 }
2813 \cs_set:Npn \thefootnote {
2814   \lngx_counter:n { footnote }
2815 }
2816 \cs_set:Npn \thempfootnote {
2817   \lngx_counter:n { mpfootnote }
2818 }
2819 \cs_set:Npn \theequation {
2820   \lngx_counter:n { equation }
2821 }
```

Here, I define the socket `lngx/native-numbering`.

```
2822
2823 \socket_new:nn { lngx / native-numbering } { 0 }
```

strict This plug sets the numbering strictly to the main language. If used, the function `\lngx_counter:n` is changed to the respective `\xxxxcounter` command (where `xxxx` stands for the main language of the document).

```

2824
2825 \socket_new_plug:nnn { lngx / native-numbering }
2826         { strict } {
2827     \cs_gset_eq:Nc \lngx_counter:n {
2828         \tl_use:N \g_lngx_main_language_tl counter
2829     }
2830 }

```

(End of definition for *strict*. This function is documented on page II.)

logical Here, I define the **logical** plug for **lngx/native-numbering**. The mechanism is pretty similar as the one used for **strict**, but here I don't renew it to the main language counter, but instead I use the **\localecounter** command provided by the **babel** package. The counters are then printed contextually (and T_EX-logically).

```

2831
2832 \socket_new_plug:nnn { lngx / native-numbering }
2833         { logical } {
2834     \cs_gset_protected:Npn \lngx_counter:n ##1 {
2835         \localecounter { digits } { ##1 }
2836     }
2837 }

```

(End of definition for *logical*. This function is documented on page II.)

off If the **off** plug is selected, then native digits are not needed. Thus the **\lngx_counter:n** is set to the unmodified **\arabic** again.

```

2838
2839 \socket_new_plug:nnn { lngx / native-numbering } { off } {
2840     \cs_gset_eq:NN \lngx_counter:n \arabic
2841 }

```

(End of definition for *off*. This function is documented on page II.)

native numbering The three choices for the **native numbering** key, i.e., **strict**, **logical** and **off** are defined here. All of them activate the plugs of their name with the **lngx/native-numbering** socket.

```

2842
2843 \cs_generate_variant:Nn \socket_assign_plug:nn { ne }
2844
2845 \keys_define:nn { lngx_keys } {
2846     native~ numbering
2847     .choices:nn          = { strict,logical,off } {
2848         \socket_assign_plug:ne { lngx / native-numbering } {
2849             \str_use:N \l_keys_choice_str
2850         }
2851         \socket_use:n { lngx / native-numbering }
2852     },

```

Similarly, we set the default value to on.

```

2853     native~ numbering
2854     .default:n           = { strict }
2855 }

```

(End of definition for *native numbering*. This function is documented on page 10.)

`\lngx_misc_reset:` Despite having sufficient control with the two plugs, there are some additional settings required by some languages that are often not needed by most others. E.g., Marathi renews the way enumerated lists are printed and that is supposed to be renewed when the language is changed. I provide a shorthand to be used for resetting such settings. It can be used in the packages of languages that don't need special settings.

```

2856
2857 \cs_new_protected:Npn \lngx_misc_reset: {
2858   \cs_set:Npn \theenumii { \alph { enumii } }
2859   \cs_set:Npn \labelenumii { ( \theenumii ) }
2860   \cs_set:Npn \theenumiii { \roman { enumiii } }
2861   \cs_set:Npn \labelenumiii { \theenumiii . }
2862   \cs_set:Npn \theenumiv { \Alph { enumiv } }
2863   \cs_set:Npn \labelenumiv { \theenumiv . }
2864   \IfPackageLoadedT { expex } {
2865     \lingset { alpha }
2866   }
2867   \cs_gset_eq:NN \emph \textit
2868 }

```

(End of definition for `\lngx_misc_reset:`. This function is documented on page 16.)

Here, I write a message to be issued when user loads an unsupported language.

```

2869
2870 \msg_new:nnn { linguistix-languages } { no_ldf } {
2871   '#1'~ is~ not~ supported.\\
2872   If~ you~ want~ it~ to~ be~ supported,~ please~ report~
2873   to~ the~ maintainers.
2874 }

```

languages I use the `.code:n` type for developing the `languages` key.

```

2875
2876 \keys_define:nn { lngx_keys } {
2877   languages
2878   .code:n          = {

```

I pass the argument of this key to a global `clist`. It is stored for public use.

```

2879   \clist_gset:Nn \g_lngx_languages_clist { #1 }

```

Since this is a public `clist` for accessing the names of the languages, I copy it to a temporary one so that the items of public interest are not lost during the operations.

```

2880   \clist_set_eq:NN \l_tmpa_clist \g_lngx_languages_clist

```

I check if the `clist` is empty or not. If it is empty, that means the user used the key without a value. In that case, `babel` already loads an 'info'-message saying that no language is loaded. So we ignore the branch and silently move to the false branch.

```

2881   \clist_if_empty:NF \l_tmpa_clist {

```

In the false branch, I pop out the first element from the `clist` to `\l_tmpa_tl`. This is the first language passed by the user. In `LINGUISTIX-LANGUAGES`, I assume that it is intended to be the first language. It is important to pop the element out because the settings used for the main language are different than the ones used for other languages.

```

2882   \clist_pop:NN \l_tmpa_clist \l_tmpa_tl

```

Since this `tl` stores the language that is going to be the main one, I equate it to another public `tl` that I will be using later in language files.

```

2883   \tl_set_eq:NN \g_lngx_main_language_tl \l_tmpa_tl

```

In `\l_tmpb_tl`, I save the options that need to go with the language stored in `\l_tmpa_tl`. The package used to have `onchar` option loaded conditionally with `LuaATeX`, but to avoid potential clashes, now it has moved to the individual package files of languages. Now I directly load the `main` option which makes the concerned language the ‘main’ language of the document.

```
2884     \tl_set:Nc \l_tmpb_tl {
2885         main,
```

To load the data from ini files, I use the `import` parameter.

```
2886         import
2887     }
```

I use the `\babelprovide` wrapper we saw earlier with the values of the first language.

```
2888     \lngx_languages:VV \l_tmpb_tl \l_tmpa_tl
```

I scan if the package for this language is available. If it is, it is loaded.

```
2889     \file_if_exist:nTF { linguistix - \l_tmpa_tl . sty } {
2890         \exp_args:Nc \RequirePackage
2891             { linguistix - \l_tmpa_tl }
2892     } {
```

If it is not, I issue the `no_ldf` warning message. It takes one argument that is the name of the language. It is extracted using the `V` argument type.

```
2893         \msg_warning:nnV { linguistix-languages } { no_ldf }
2894                                     \l_tmpa_tl
2895     }
```

The temporary `tl`s are cleared.

```
2896     \tl_clear:N \l_tmpa_tl
2897     \tl_clear:N \l_tmpb_tl
```

I again check if the `clist` is empty. If it is, it means the user is typesetting a monolingual document as they don’t need any other language than the ‘main’ one.

```
2898     \clist_if_empty:NF \l_tmpa_clist {
```

Now I have to repeat the same actions for all the pending languages. I do it with `\clist_map_inline:Nn`.

```
2899         \clist_map_inline:Nn \l_tmpa_clist {
2900             \clist_pop:NN \l_tmpa_clist \l_tmpa_tl
2901             \tl_set:Nc \l_tmpb_tl { import }
2902             \lngx_languages:VV \l_tmpb_tl \l_tmpa_tl
2903             \file_if_exist:nTF {
2904                 linguistix - \l_tmpa_tl . sty
2905             } {
2906                 \exp_args:Nc \RequirePackage
2907                     { linguistix - \l_tmpa_tl }
2908             } {
2909                 \msg_warning:nnV { linguistix-languages }
2910                     { no_ldf }
2911                     \l_tmpa_tl
2912             }
2913             \tl_clear:N \l_tmpa_tl
2914             \tl_clear:N \l_tmpb_tl
2915         }
2916     }
2917 }
```

```
2918     }  
2919   }  
2920 </lang>
```

*(End of definition for **languages**. This function is documented on page **10**.)*

```

2921 <*logos>
2922 \ProvidesExplPackage{linguistix-logos}
2923         {2025-12-22}
2924         {v0.6a}
2925         {%
2926         Logos of the ‘LinguisTiX’ bundle.%
2927         }

```

The fontspec package (if not already loaded).

```

2928
2929 \IfPackageLoadedF { fontspec } {
2930 \RequirePackage { fontspec }
2931 }

```

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

```

2932
2933 \newfontfamily \lngx_logo_font: [
2934   IgnoreFontspecFile,
2935   UprightFont          = { NewCMUncial10-Book.otf },
2936   UprightFeatures      = {
2937     SizeFeatures       = {
2938       {
2939         Size           = {-8},
2940         Font           = {NewCMUncial08-Book.otf}
2941       },
2942       {
2943         Size           = {8-},
2944         Font           = {NewCMUncial10-Book.otf}
2945       },
2946     }
2947   },
2948   BoldFont              = { NewCMUncial10-Bold.otf },
2949   BoldFeatures          = {
2950     SizeFeatures       = {
2951       {
2952         Size           = {-8},
2953         Font           = {NewCMUncial08-Bold.otf}
2954       },
2955       {
2956         Size           = {8-},
2957         Font           = {NewCMUncial10-Bold.otf}
2958       },
2959     }
2960   },
2961   Renderer              = { HarfBuzz }
2962 ]{ NewCMUncial10-Book.otf }

```

(End of definition for \lngx_logo_font:. This function is documented on page 16.)

lngx_purple_color The following defines the lngx_purple_color.

```

2963
2964 \color_set:nn { lngx_purple_color } { blue ! 50 ! red }

```

(End of definition for lngx_purple_color. This function is documented on page 16.)

\lngxlogo Here, I define the commands for printing various logos.

```
2965
2966 \NewDocumentCommand \lngxlogo { 0{} } {%
2967   \group_begin:
2968   \lngx_logo_font:
2969   LinguisTi
2970   \color_group_begin:
2971   \color_select:n { lngx_purple_color }
2972   X
2973   \color_group_end:
2974   \IfBlankF { #1 } { - #1 }
2975   \group_end:
2976 }
```

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

\lngxpkg Since we need expandable commands, I use the non-protected function, **\cs_new:Npn** for defining them.

```
2977
2978 \cs_new:Npn \lngxpkg {
2979   \IfPackageLoadedTF { hyperref } {
2980     \texorpdfstring {
2981       \lngxlogo
2982     } {
2983       LinguisTiX
2984     }
2985   } {
2986     \lngxlogo
2987   }
2988 }
```

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

\lngxbaselogo Here, I define all the logos with a **clist**. The package names are stored in the **clist** and then used at appropriate positions.

```
\lngxfontslogo
\lngxipalogo
\lngxlogoslogo
\lngxnfsslogo
2989
2990 \clist_map_inline:nn {
2991   base,
2992   examples,
2993   fixpex,
2994   fonts,
2995   ipa,
2996   languages,
2997   logos,
2998   nfss,
2999   marathi,
3000   british,
3001   american,
3002   english,
3003   greek
3004 } {
```

#1 is substituted with the package name. First, for the command-name itself, then as the optional argument of **\lngxlogo** and then in the PDF-string.

```

3005 \cs_new:cpn { lngx #1 logo } {
3006   \texorpdfstring {
3007     \lngxlogo [ #1 ]
3008   } {
3009     LinguisTiX - #1
3010   }
3011 }
3012 }
3013 </logos>

```

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

LINGUIS*TiX*-NFSS

Documentation | [L^AT_EX₃-interface](#)

```

3014 < *nfss >
3015 \ProvidesExplPackage{linguistix-nfss}
3016   {2025-12-22}
3017   {v0.6a}
3018   {%
3019     An extension to the core NFSS commands
3020     from the ‘LinguisTiX’ bundle.%
3021   }

```

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.

```

3022
3023 \tl_new:N \l__lngx_normalfont_tmp_tl
3024 \tl_new:N \l__lngx_selectfont_tmp_tl
3025 \tl_new:N \l__lngx_family_tmp_tl
3026 \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.

```

3027
3028 \tl_new:N \l_lngx_current_encoding_tl
3029 \tl_new:N \l_lngx_current_meta_family_tl
3030 \tl_new:N \l_lngx_current_super_family_tl
3031 \tl_new:N \l_lngx_current_series_tl
3032 \tl_new:N \l_lngx_current_shape_tl

```

`\c_lngx_default_rmdefault_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_sfdefault_tl`

`\c_lngx_default_ttdefault_tl`

```

3033
3034 \hook_gput_code:nnn { begindocument / end } { . } {
3035   \tl_const:Ne \c_lngx_default_rmdefault_tl { \rmdefault }
3036   \tl_const:Ne \c_lngx_default_sfdefault_tl { \sfdefault }
3037   \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 [I7](#).)

`\l_lngx_current_encoding_tl`
`\l_lngx_current_meta_family_tl`
`\l_lngx_current_super_family_tl`
`\l_lngx_current_series_tl`
`\l_lngx_current_shape_tl`

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

```

3038 \tl_set:Nn \l_lngx_current_super_family_tl { default }

```

The current encoding is saved in the relevant `tl`.

```
3039 \tl_set:Ne \l_lngx_current_encoding_tl {
3040   \encodingdefault
3041 }
```

When the package was first released, there was no public interface for guessing the current meta family, but from `ltnews42`, `\@currentmetafamily` became available. Thanks Frank for pointing this out.

```
3042 \tl_set:Ne \l_lngx_current_meta_family_tl {
3043   \@currentmetafamily % new from ltnews42, thanks Frank!
3044 }
```

Here, the series and shape `tl`s are set to their defaults.

```
3045 \tl_set:Nn \l_lngx_current_series_tl { md }
3046 \tl_set:Nn \l_lngx_current_shape_tl { up }
3047 }
```

(End of definition for `\l_lngx_current_encoding_tl` and others. These functions are documented on page 17.)

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

```
3048
3049 \hook_gput_code:nnn { cmd / selectfont / before } { . } {
3050   \tl_set:Ne \l__lngx_selectfont_tmp_tl { \f@encoding }
3051 }
```

After the processing of `\selectfont`, I equate the temporary `tl` with the one that the package is tracking. This way, the effect of `\selectfont` 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 `\selectfont`.

```
3052
3053 \hook_gput_code:nnn { cmd / selectfont / after } { . } {
3054   \tl_set_eq:NN \l_lngx_current_encoding_tl
3055     \l_lngx_selectfont_tmp_tl
3056   \tl_clear:N \l__lngx_selectfont_tmp_tl
3057 }
```

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

```
3058
3059 \hook_gput_code:nnn { cmd / rmfamily / before } { . } {
3060   \tl_set:Nn \l_lngx_current_meta_family_tl { rm }
3061   \tl_set:Ne \l__lngx_family_tmp_tl { \f@encoding }
3062 }
3063
3064 \hook_gput_code:nnn { cmd / rmfamily / after } { . } {
3065   \tl_set:Nn \l_lngx_current_meta_family_tl { rm }
3066   \tl_set_eq:NN \l_lngx_current_encoding_tl
3067     \l__lngx_family_tmp_tl
3068   \tl_clear:N \l__lngx_family_tmp_tl
3069 }
3070
3071 \hook_gput_code:nnn { cmd / sffamily / before } { . } {
3072   \tl_set:Nn \l_lngx_current_meta_family_tl { sf }
```

```

3073 \tl_set:Nn \l__lngx_family_tmp_tl { \f@encoding }
3074 }
3075
3076 \hook_gput_code:nnn { cmd / sffamily / after } { . } {
3077   \tl_set:Nn \l_lngx_current_meta_family_tl { sf }
3078   \tl_set_eq:NN \l_lngx_current_encoding_tl
3079     \l__lngx_family_tmp_tl
3080   \tl_clear:N \l__lngx_family_tmp_tl
3081 }
3082
3083 \hook_gput_code:nnn { cmd / ttfamily / before } { . } {
3084   \tl_set:Nn \l_lngx_current_meta_family_tl { tt }
3085   \tl_set:Nn \l__lngx_family_tmp_tl { \f@encoding }
3086 }
3087
3088 \hook_gput_code:nnn { cmd / ttfamily / after } { . } {
3089   \tl_set:Nn \l_lngx_current_meta_family_tl { tt }
3090   \tl_set_eq:NN \l_lngx_current_encoding_tl
3091     \l__lngx_family_tmp_tl
3092   \tl_clear:N \l__lngx_family_tmp_tl
3093 }

```

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

```

3094
3095 \hook_gput_code:nnn { cmd / mdseries / after } { . } {
3096   \tl_set:Nn \l_lngx_current_series_tl { md }
3097 }
3098
3099 \hook_gput_code:nnn { cmd / bfseries / after } { . } {
3100   \tl_set:Nn \l_lngx_current_series_tl { bf }
3101 }

```

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

```

3102
3103 \hook_gput_code:nnn { cmd / upshape / after } { . } {
3104   \tl_set:Nn \l_lngx_current_shape_tl { up }
3105 }
3106
3107 \hook_gput_code:nnn { cmd / itshape / after } { . } {
3108   \tl_set:Nn \l_lngx_current_shape_tl { it }
3109 }
3110
3111 \hook_gput_code:nnn { cmd / scshape / after } { . } {
3112   \tl_set:Nn \l_lngx_current_shape_tl { sc }
3113 }
3114
3115 \hook_gput_code:nnn { cmd / sscshape / after } { . } {
3116   \tl_set:Nn \l_lngx_current_shape_tl { ssc }
3117 }
3118
3119 \hook_gput_code:nnn { cmd / slshape / after } { . } {
3120   \tl_set:Nn \l_lngx_current_shape_tl { sl }

```



```

3121 }
3122
3123 \hook_gput_code:nnn { cmd / swshape / after } { . } {
3124   \tl_set:Nn \l_lngx_current_shape_tl { sw }
3125 }
3126
3127 \hook_gput_code:nnn { cmd / ulcshape / after } { . } {
3128   \tl_set:Nn \l_lngx_current_shape_tl { ulc }
3129 }

```

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

```

3130
3131 \prg_new_conditional:Nnn \lngx_if_encoding:n {
3132   P,
3133   T,
3134   F,
3135   TF
3136 } {
3137   \tl_if_eq:NnTF \l_lngx_current_encoding_tl { #1 } {
3138     \prg_return_true:
3139   } {
3140     \prg_return_false:
3141   }
3142 }
3143

```

(End of definition for `\lngx_if_encoding:nTF`. This function is documented on page 17.)

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

```

3144
3145 \cs_new_eq:NN \IfEncodingTF \lngx_if_encoding:nTF
3146 \cs_new_eq:NN \IfEncodingT \lngx_if_encoding:nT
3147 \cs_new_eq:NN \IfEncodingF \lngx_if_encoding:nF

```

(End of definition for `\IfEncodingTF`, `\IfEncodingT`, and `\IfEncodingF`. These functions are documented on page 19.)

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

```

3148
3149 \prg_new_conditional:Nnn \lngx_if_meta_family:n {
3150   P,
3151   T,
3152   F,
3153   TF
3154 } {
3155   \tl_if_eq:NnTF \l_lngx_current_meta_family_tl { #1 } {
3156     \prg_return_true:
3157   } {
3158     \prg_return_false:
3159   }
3160 }

```

(End of definition for `\lngx_if_meta_family:nTF`. This function is documented on page 17.)

\IfMetaFamilyTF User-facing conditionals for meta family.

\IfMetaFamilyT
\IfMetaFamilyF

```

3161
3162 \cs_new_eq:NN \IfMetaFamilyTF \lngx_if_meta_family:nTF
3163 \cs_new_eq:NN \IfMetaFamilyT \lngx_if_meta_family:nT
3164 \cs_new_eq:NN \IfMetaFamilyF \lngx_if_meta_family:nF

```

(End of definition for `\IfMetaFamilyTF`, `\IfMetaFamilyT`, and `\IfMetaFamilyF`. These functions are documented on page 13.)

\lngx_if_super_family_p:n
\lngx_if_super_family:nTF

A conditional for checking the super family with the given argument.

```

3165
3166 \prg_new_conditional:Nnn \lngx_if_super_family:n {
3167   P,
3168   T,
3169   F,
3170   TF
3171 } {
3172   \tl_if_eq:NnTF \l_lngx_current_super_family_tl { #1 } {
3173     \prg_return_true:
3174   } {
3175     \prg_return_false:
3176   }
3177 }

```

(End of definition for `\lngx_if_super_family:nTF`. This function is documented on page 17.)

\IfSuperFamilyTF User-facing conditionals for super family.

\IfSuperFamilyT
\IfSuperFamilyF

```

3178
3179 \cs_new_eq:NN \IfSuperFamilyTF \lngx_if_super_family:nTF
3180 \cs_new_eq:NN \IfSuperFamilyT \lngx_if_super_family:nT
3181 \cs_new_eq:NN \IfSuperFamilyF \lngx_if_super_family:nF

```

(End of definition for `\IfSuperFamilyTF`, `\IfSuperFamilyT`, and `\IfSuperFamilyF`. These functions are documented on page 13.)

\lngx_if_series_p:n
\lngx_if_series:nTF

A conditional for checking the current series with the given argument.

```

3182
3183 \prg_new_conditional:Nnn \lngx_if_series:n {
3184   P,
3185   T,
3186   F,
3187   TF
3188 } {
3189   \tl_if_eq:NnTF \l_lngx_current_series_tl { #1 } {
3190     \prg_return_true:
3191   } {
3192     \prg_return_false:
3193   }
3194 }

```

(End of definition for `\lngx_if_series:nTF`. This function is documented on page 17.)

`\IfSeriesTF` Its user-side macros.

`\IfSeriesT`

`\IfSeriesF`

```
3195
3196 \cs_new_eq:NN \IfSeriesTF \lngx_if_series:nTF
3197 \cs_new_eq:NN \IfSeriesT \lngx_if_series:nT
3198 \cs_new_eq:NN \IfSeriesF \lngx_if_series:nF
```

(End of definition for `\IfSeriesTF`, `\IfSeriesT`, and `\IfSeriesF`. These functions are documented on page 13.)

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

`\lngx_if_shape:nTF`

```
3199
3200 \prg_new_conditional:Nnn \lngx_if_shape:n {
3201   P,
3202   T,
3203   F,
3204   TF
3205 } {
3206   \tl_if_eq:NnTF \l_lngx_current_shape_tl { #1 } {
3207     \prg_return_true:
3208   } {
3209     \prg_return_false:
3210   }
3211 }
```

(End of definition for `\lngx_if_shape:nTF`. This function is documented on page 17.)

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

`\IfShapeT`

`\IfShapeF`

```
3212
3213 \cs_new_eq:NN \IfShapeTF \lngx_if_shape:nTF
3214 \cs_new_eq:NN \IfShapeT \lngx_if_shape:nT
3215 \cs_new_eq:NN \IfShapeF \lngx_if_shape:nF
```

(End of definition for `\IfShapeTF`, `\IfShapeT`, and `\IfShapeF`. These functions are documented on page 13.)

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.

```
3216
3217 \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`

```
3218
3219 \clist_map_inline:nn {
3220   rm,
3221   sf,
3222   tt
3223 } {
3224   \prg_new_conditional:cnn { lngx_if_meta_family_ #1 : } {
3225     p, T, F, TF
3226   } {
3227     \tl_if_eq:NnTF \l_lngx_current_meta_family_tl { #1 } {
3228       \prg_return_true:
3229     } {
3230       \prg_return_false:
```

```

3231     }
3232   }
3233 }

```

(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 17.)

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

```

\lngx_if_series_md:TF
\lngx_if_series_bf_p:
\lngx_if_series_bf:TF
3234
3235 \clist_map_inline:nn {
3236   md,
3237   bf
3238 } {
3239   \prg_new_conditional:cnn { lngx_if_series_ #1 : } {
3240     p, T, F, TF
3241   } {
3242     \tl_if_eq:NnTF \l_lngx_current_series_tl { #1 } {
3243       \prg_return_true:
3244     } {
3245       \prg_return_false:
3246     }
3247   }
3248 }

```

(End of definition for `\lngx_if_series_md:TF` and `\lngx_if_series_bf:TF`. These functions are documented on page 17.)

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

```

\lngx_if_shape_up:TF
\lngx_if_shape_it_p:
\lngx_if_shape_it:TF
\lngx_if_shape_sc_p:
\lngx_if_shape_sc:TF
\lngx_if_shape_ssc_p:
\lngx_if_shape_ssc:TF
\lngx_if_shape_sl_p:
\lngx_if_shape_sl:TF
\lngx_if_shape_sw_p:
\lngx_if_shape_sw:TF
\lngx_if_shape_ulc_p:
\lngx_if_shape_ulc:TF
3249
3250 \clist_map_inline:nn {
3251   up,
3252   it,
3253   sc,
3254   ssc,
3255   sl,
3256   sw,
3257   ulc
3258 } {
3259   \prg_new_conditional:cnn { lngx_if_shape_ #1 : } {
3260     p, T, F, TF
3261   } {
3262     \tl_if_eq:NnTF \l_lngx_current_shape_tl { #1 } {
3263       \prg_return_true:
3264     } {
3265       \prg_return_false:
3266     }
3267   }
3268 }

```

(End of definition for `\lngx_if_shape_up:TF` and others. These functions are documented on page 18.)

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 `tl`s 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.

```

3269
3270 \clist_map_inline:nn {
3271     rm,
3272     sf,
3273     tt
3274 } {
3275     \keys_define:nn { lngx_nfss } {
3276         #1
3277         .code:n          = {
3278             \tl_gclear_new:c {
3279                 g_lngx_ \l__lngx_nfss_tmp_tl _ #1 default _tl
3280             }
3281             \tl_gset:cn {
3282                 g_lngx_ \l__lngx_nfss_tmp_tl _ #1 default _tl
3283             } { ##1 }
3284         }
3285     }
3286 }

```

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

```

3287
3288 \cs_new_protected:Npn \lngx_super_font_family:nn #1#2 {
3289     \tl_set:Nx \l__lngx_nfss_tmp_tl { #1 }
3290
3291     Now, I pass the second argument to the key-set I just defined. The temporary tl is
3292     cleared. This function comes with a user-side macro.
3293
3294     \keys_set:nn { lngx_nfss } { #2 }
3295     \tl_clear:N \l__lngx_nfss_tmp_tl
3296 }
3297
3298 \cs_gset_eq:NN \superfontfamily
3299 \lngx_super_font_family:nn

```

(End of definition for `\lngx_super_font_family:nn` and `\superfontfamily`. These functions are documented on page 18.)

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

```

3296
3297 \cs_new_protected:Npn \lngx_soft_super_font_family:nn #1#2 {
3298     \tl_set:Nx \l_lngx_current_super_family_tl { #1 }
3299
3300     I first check if the tls for rm, sf and tt are empty or not. Only if they are not, I use their
3301     content in the respective \XXdefault. This makes the use of all the keys optional. Only
3302     the keys that the user has used are processed here.
3303
3304     \clist_map_inline:nn {
3305         rm,
3306         sf,
3307         tt
3308     } {
3309         \tl_if_empty:cF { g_lngx_ #1 _ ##1 default_tl } {
3310             \cs_set:cpe { ##1 default } {
3311                 \tl_use:c { g_lngx_ #1 _ ##1 default _tl }
3312             }
3313         }
3314     }
3315 }

```

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

```
3310 \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.

```
3311 \clist_map_inline:nn { #2 } {
3312   \str_case:nn { ##1 } {
3313     { encoding } {
3314       \exp_args:NV \fontencoding
3315         \l_lngx_current_encoding_tl
3316     }
3317     { family } {
3318       \use:c {
3319         \l_lngx_current_meta_family_tl family
3320       }
3321       \exp_args:NV \fontencoding
3322         \l_lngx_current_encoding_tl
3323       \selectfont
3324     }
3325     { series } {
3326       \use:c {
3327         \l_lngx_current_series_tl series
3328       }
3329     }
3330     { shape } {
3331       \use:c {
3332         \l_lngx_current_shape_tl shape
3333       }
3334     }
3335   }
3336 }
3337 }
3338
3339 \cs_gset_eq:NN \softsuperfontfamily
3340   \lngx_soft_super_font_family:nn
```

(End of definition for `\lngx_soft_super_font_family:nn` and `\softsuperfontfamily`. These functions are documented on page 18.)

```
\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.

```
3341
3342 \cs_new_protected:Npn \lngx softer_super_font_family:n #1 {
3343   \lngx_soft_super_font_family:nn { #1 } {
3344     family,
3345     series,
3346     shape
3347   }
3348 }
3349
3350 \cs_gset_eq:NN \softsuperfontfamily
3351   \lngx softer_super_font_family:n
```

(End of definition for `\lngx softer super font family:n` and `\softersuperfontfamily`. These functions are documented on page 18.)

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

```

3352
3353 \cs_new_protected:Npn \lngx_softest_super_font_family:n #1 {
3354   \lngx_soft_super_font_family:nn { #1 } {
3355     encoding,
3356     family,
3357     series,
3358     shape
3359   }
3360 }
3361
3362 \cs_gset_eq:NN \softestsuperfontfamily
3363   \lngx_softest_super_font_family:n

```

(End of definition for `\lngx_softest_super_font_family:n` and `\softestsuperfontfamily`. These functions are documented on page 18.)

`\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.

```

3364
3365 \cs_new_protected:Npn \lngx_soft_normal_font:n #1 {
3366   \tl_set:Nc \l_lngx_current_super_family_tl { default }
3367   \clist_map_inline:nn {
3368     rm,
3369     sf,
3370     tt
3371   } {
3372     \cs_set:cpe { ##1 default } {
3373       \tl_use:c { c_lngx_default_ ##1 default _tl }
3374     }
3375   }
3376   \normalfont
3377   \clist_map_inline:nn { #1 } {
3378     \str_case:nn { ##1 } {
3379       { encoding } {
3380         \exp_args:NV \fontencoding
3381           \l_lngx_current_encoding_tl
3382       }
3383       { family } {
3384         \use:c {
3385           \l_lngx_current_meta_family_tl family
3386         }
3387         \exp_args:NV \fontencoding
3388           \l_lngx_current_encoding_tl
3389         \selectfont
3390       }
3391       { series } {
3392         \use:c {
3393           \l_lngx_current_series_tl series
3394         }
3395       }

```

```

3396     { shape } {
3397         \use:c {
3398             \l_lngx_current_shape_tl shape
3399         }
3400     }
3401 }
3402 }
3403 }
3404
3405 \cs_gset_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 18.)

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

```

3406
3407 \cs_new_protected:Npn \lngx_softer_normal_font: {
3408     \lngx_soft_normal_font:n {
3409         family,
3410         series,
3411         shape
3412     }
3413 }
3414
3415 \cs_gset_eq:NN \softernormalfont \lngx_softer_normal_font:

```

(End of definition for `\lngx_softer_normal_font:` and `\softernormalfont`. These functions are documented on page 18.)

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

```

3416
3417 \cs_new_protected:Npn \lngx_softest_normal_font: {
3418     \lngx_soft_normal_font:n {
3419         encoding,
3420         family,
3421         series,
3422         shape
3423     }
3424 }
3425
3426 \cs_gset_eq:NN \softestnormalfont \lngx_softest_normal_font:

```

(End of definition for `\lngx_softest_normal_font:` and `\softestnormalfont`. These functions are documented on page 18.)

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

```

3427 \cs_new:Npn \CurrentEncoding {
3428     \tl_use:N \l_lngx_current_encoding_tl
3429 }
3430 \cs_new:Npn \CurrentMetaFamily {
3431     \tl_use:N \l_lngx_current_meta_family_tl
3432 }
3433 \cs_new:Npn \CurrentSuperFamily {
3434     \tl_use:N \l_lngx_current_super_family_tl

```



```

3435 }
3436 \cs_new:Npn \CurrentSeries {
3437   \tl_use:N \l_lngx_current_series_tl
3438 }
3439 \cs_new:Npn \CurrentShape {
3440   \tl_use:N \l_lngx_current_shape_tl
3441 }
3442 \</nfss>

```

(End of definition for `\CurrentEncoding` and others. These functions are documented on page [I3](#).)

References

- Bringinghurst, Robert (2004). *The elements of typographic style*. 4th ed. Point Roberts, WA: Hartley & Marks, Publishers.
- Munn, Alan and Enrico Gregorio (5th Dec. 2023). *ExPex fails with unicode-math. How to avoid the clash?* URL: <https://tex.stackexchange.com/q/703094> (visited on 21/12/2025).

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. Secondarily, 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.