

The LINGUISṬIX bundle

निरंजन

14 May 2026 (v1.0)

🏠 <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	11
2	Funding	4		Interface... 21; Implementation... 56	
3	Acknowledgements	4	9	LINGUISṬIX-LANGUAGES	14
4	LINGUISṬIX-BASE	5		Interface... 21; Implementation... 65	
	Interface... 19; Implementation... 25		10	LINGUISṬIX-LOGOS	16
5	LINGUISṬIX-FIXPEX	5		Interface... 22; Implementation... 72	
	Interface... 20; Implementation... 26		11	LINGUISṬIX-NFSS	17
6	LINGUISṬIX-FONTS	5		Interface... 22; Implementation... 73	
	Interface... 20; Implementation... 28				
7	LINGUISṬIX-eLOSSING	8		Index	86
	Interface... 20; Implementation... 40				

The LINGUISṬIX bundle

Copyright © 2025, 2026 निरंजन

This program is free software: you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.

This program is distributed in the hope that it will be useful, but *without any warranty*; without even the implied warranty of *merchantability* or *fitness for a particular purpose*. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with this program. If not, see <https://www.gnu.org/licenses/>.

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. If you are an impatient reader and are just willing to read the users manual, you may skip reading the current section and start with section 4 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 [ɑ], but unfortunately, in Italic shape, the latter is a variant of the former. E.g., `[a\textit{a}]` produces '[aa]'. Whenever an author uses Italic shape for their transcription and use `a`, a wrong IPA symbol is printed with most fonts. This problem was kindly acknowledged by Antonis Tzolomitis, the developer of New Computer Modern. In the stylistic set dedicated for linguistics, the correct shape was added to the Italic shape by him. Thus, `\ipatext{a\textit{a}}` (a command from `LINGUIS\X-ipa`) renders '[aa]'. The package enables New Computer Modern family with stylistic set `o5` dedicated for IPA. It also adds the brackets or slashes around the argument as explained in section 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 [g̱g̱].

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

2 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``TLX` 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).

An experimental version of `LINGUIS``TLX`-`eGLOSSING` is released on 2026-01-19. This version is for testing and getting feedback from the community. This marks the completion of the first grant provided by the `TeX` users group's. The project will still continue to develop further, so funding initiatives will be highly appreciated.

3 Acknowledgements

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

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

Throughout the development of `LINGUIS``TLX`-`eGLOSSING`, Shireen Irani helped me with her valuable comments regarding the accessibility needs in the field of linguistics. I thank her for her constant 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``TLX` 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.

4 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:

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

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

5 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 `(lua)-unicode-math`. It provides a single command.

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

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

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

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

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

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

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

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

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

Note that the colour of the available options for the `old style numbers` key is yellow. It is on purpose like that to suggest that the default selection of its truth value is not unconditional. Let’s look into why it is so.

Old style numbers are among the peculiar features of the Latin script. It has many letters that have ascenders and many that have descenders. Although, this is observed in the Latin script, it is not a robust feature seen in all the scripts. E.g., Cyrillic script has very sparse number of them or scripts like Japanese do not have any of them at all. The languages that use non-Latin script seem to prefer to avoid old style numbers as they may create an obstacle in the natural reading flow. This was brought to my notice by Alexey Kuznetsov while developing the language package of Russian and I thank him very much for raising this important issue. Starting from version 0.9c, old style numbers are *not* set by default. But this is mostly not a breaking change, at least for English documents. They still are set globally with the English language packages. They also are set if `LINGUISŒX-LANGUAGES` package is not loaded till the end of the document. So, if you were just using `LINGUISŒX-FONTS` package, there is no change for you, but if your document was written in any other language supported by `LINGUISŒX`, then you won’t see the default old style numbers that you used to see prior to this version. This is why the colour of the available options is yellow as the default value is not fixed.

Note that printing 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. Otherwise you may get a (harmless) warning.

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
0123456789	Old style with the old 1
0123456789	Lining

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

2 Maths

LINGUISTX-FONTS sets maths fonts also. I have used `lua-unicode-math` package which is faster and which is said to be the future of maths in \LaTeX . But, as of now it is highly experimental. If you want to stick to the stable `unicode-math` package. The trick is simply to load the same before loading LINGUISTX. That will suppress the loading of `lua-unicode-math`. 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.

<code>bourbaki's empty set</code>	<code>= {\langle truth value \rangle}</code>	<code>true false</code>
-----------------------------------	--	---------------------------

In $(\mathbb{A})\text{\TeX}$, the default shape of the ‘empty set’ symbol is: ‘ \emptyset ’, but the symbol used by the Bourbaki group is still considered more correct and preferred by many (including me). New Computer Modern Math fonts provide it by default and the slashed zero is provided as a character variant. Since the Unicode-correct `\emptyset` is activated by the package, it always renders: ‘ \emptyset ’ and not: ‘ \emptyset ’. In order to change this behaviour, one may use this key and set it to false for getting the slashed-zero of original $(\mathbb{A})\text{\TeX}$. Hail plumbers union, *IYKYK!* ;-)

This package provides a suit for creating interlinear glosses. It is supported by T_EX users group's devfund. The package attempts to be an all-in-one solution for glossing. It doesn't provide any particular glosses. It only provides a method to create them. Using it, one may easily create packages like LINGUIS $\overline{\text{T}}$ X-Leipzig to support a set of glosses. The glosses created by the package use the new code of the L^AT_EX project as they are created in a tagging aware manner. Each gloss sets a hyperlink to its position in the list of glosses. Let's take a look at its commands and options.

\backslash glx	{ <i>comma separated list of glosses</i> }
\backslash glx*	{ <i>comma separated list of glosses</i> }

These simple commands take a comma separated list as their argument. All the items from the list are glosses (either created by the user or provided by a package). Cases of the items given in the list are ignored. Spaces around the items are ignored. The regular unstarred command prints the glosses related to each of the item in the comma separated list, whereas the starred variant prints their expansions. Have a look at the following example.

```

\DocumentMetadata{tagging=on,lang={en-GB}}
\documentclass{article}
\usepackage{linguistix}

\begin{document}
\glx{prs,pst}\par
\glx{ prs, pst  }\par
\glx{ Prs,pSt}

\glx*{prs,pst}\par
\glx*{ prs, pst  }\par
\glx*{ Prs,pST}
\end{document}

```

The expansions of PST and PRS (from LINGUIS $\overline{\text{T}}$ X-Leipzig package) are past and present respectively. This example produces identical output in three lines for glosses and the same for its expansions. Notice that there is no format to the cases of the glosses and similarly one level of spaces are trimmed.

\backslash newgloss	{ <i>gloss</i> } { <i>expansion</i> }
\backslash renewgloss	{ <i>gloss</i> } { <i>expansion</i> }

These commands create a new gloss or renew an existing one. They can be accessed with the \backslash glx command as explained above. Using \backslash renewgloss mid-document is not recommended as it will erase the data of page numbers for the previous (renewed) version of it.

\backslash listofglosses	[<i>setup keys</i>]
----------------------------	-----------------------

This command prints the list of glosses using the default settings. If the optional argument is used, the adjustments are made locally only for a single run. E.g.:

Glossary

PRS: present 8 | PST: past 8

`\setupglossing` *{(keys for formatting glosses)}*

This command takes one argument, i.e., the keys that control everything regarding the use of glosses and their expansions. The keys it takes are described in the section that follows.

1 Setting up the glosses

The following keys can be passed to the command `\setupglossing`. They control the printing along with a lot of other things regarding glosses. All the customisation offered by the package can be accessed via this command.

`format` = *{(formatted element gloss/expansion)}* `gloss | expansion`

The `format` key is used for setting the format of either the gloss or the expansion. It's a meta key that takes other key-val pair in the argument. The nested keys control the formatting of the respective elements.

`gloss` = *{(formatting commands for glosses)}* `\textsc{#1}`
`expansion` = *{(formatting commands for glosses)}*

These keys only work inside the meta key `format`. They set the commands that print either the gloss or the expansion. `#1` refers to the printed text of them. No special formatting is applied to expansions by default, but glosses are by default printed in `\textsc`.

`link color` = *{(link color)}* `black`

This option locally sets the colour for the hyperlinks. By default they are set to the black colour.

`sort` = *{(sorting style)}* `alphabetical | use`

This key controls how the keys printed in the list of glosses are ordered. They may be ordered alphabetically or following the sequence in which they were used, the former being the default.

`expansion case` = *{(case)}* `lowercase | title case all | title case first`

The expansion can be printed in one of these three cases. The default printing happens in lowercase.

`style` = *{(glossary style)}* `block | inline`

The package offers two styles. The `inline` style prints the glosses and their expansions without page numbers in the flowing text, whereas the `block` style, in default settings prints them in a multicolumn block with an unnumbered section with the glossary name.

<u>columns</u>	= $\{(\textit{number of columns})\}$	2
	The block style of glosses is printed in multicolumn layout by default. If the number of columns has to be adjusted, this key shall be used. The default value of it is 2. It works with only one column too.	
<u>page numbers</u>	= $\{(\textit{truth value})\}$	true false
	By default, page numbers on which a particular gloss was used are printed in the block style. This can be turned off with this bool key.	
<u>sectioning</u>	= $\{(\textit{section level})\}$	section
	In block style, a section heading is printed. In order to choose the level of sectioning, this command can be used. The default is section which can be changed to any other desired level. In addition the key allows an option null which suppresses the use of any section heading.	
<u>section number</u>	= $\{(\textit{truth value})\}$	true false
	By default, the section number for the glossary is turned off, but if one wants to print it, this bool key can be used with the true value.	
<u>no bold</u>	= $\{(\textit{truth value})\}$	true false
	Generally, the glosses are printed in bold inside glossary. Some fonts don't have bold small caps (e.g., Latin Modern). If you need to stick to them, you can use this inverse bool key with true value in order to obtain non-bold glosses.	
<u>separator</u>	= $\{(\textit{separator between glosses or expansions})\}$	
	This is a context-sensitive key. If used with <code>\glx</code> , then it sets the separator between the glosses (<code>,_</code> is the default). If used with <code>\glx*</code> , it sets the separator between the expansions (<code>,_</code> is the default) and if used with the <code>\listofglosses</code> , it sets the separator between glosses and their expansions (<code>:_</code> is the default).	
<u>entry separator</u>	= $\{(\textit{separator between pairs of glosses and expansions})\}$	
	Each pair of gloss and its expansion is separated using a token list controlled by this key. The default is <code>\par</code> .	

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

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

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

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

```
\lngxipa
```

This also is a command that switches to the IPA-only features (default as well as user added). This command, of course, leaks and that's why *should* be delimited. E.g., the following code lines produce [a_ɪ p^hi: e_ɪ] and /a_ɪ p^hi: e_ɪ/ respectively:

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

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

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

Let's now see the combined table of font keys provided by both LINGUIS $\overline{\text{C}}$ I $\overline{\text{X}}$ -FONTS and LINGUIS $\overline{\text{C}}$ I $\overline{\text{X}}$ -ipa.

Family	LINGUIS $\overline{\text{C}}$ I $\overline{\text{X}}$ -FONTS	LINGUIS $\overline{\text{C}}$ I $\overline{\text{X}}$ -ipa
Serif	text main font	ipa main font
	text upright	ipa upright
	text upright features	ipa upright features
	text bold	ipa bold
	text bold features	ipa bold features
<i>Continued on the next page...</i>		

Family	LINGUISCI _X -FONTS	LINGUISCI _X -IPA
	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 font	ipa sans font
	text sans upright	ipa sans upright
	text sans upright features	ipa sans upright features
	text sans bold	ipa sans bold
	text sans bold features	ipa sans bold 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 font	ipa mono font
	text mono upright	ipa mono upright
	text mono upright features	ipa mono upright features
	text mono bold	ipa mono bold
	text mono bold features	ipa mono bold features
	text mono italic	ipa mono italic
	text mono italic features	ipa mono italic features
	text mono bold italic	ipa mono bold italic
	text mono bold italic features	ipa mono bold italic features
	text mono slanted	ipa mono slanted
	text mono slanted features	ipa mono slanted features
	text mono bold slanted	ipa mono bold slanted
<i>Continued on the next page...</i>		

Family	LINGUISṬṬX-FONTS	LINGUISṬṬX-IPA
	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 LINGUISṬṬX-FONTS and LINGUISṬṬX-IPA

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

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

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^AT_EX, 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/linguistx-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 LINGUIST $\text{\textcolor{violet}{X}}$.

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-speakers scrutiny. Whatever is wrong in `babel`, may not get corrected in LINGUIST $\text{\textcolor{violet}{X}}$.

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., in nested `enumerate` environment, Marathi 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.

<hr/> <code>languages</code> <hr/>	<code>{\list of languages}</code>
<code>\loadlanguages</code>	<code>{\list of languages}</code>
	This key works with the central key-parser of <code>LINGUISTIX</code> , i.e., <code>\linguistix</code> . It accepts one argument that is a list of languages user wants to load. Unlike <code>babel</code> , the first element of this list is set as the main language for the document. The command <code>\loadlanguages</code> has the identical behaviour. In fact, it is a wrapper around the key.
<hr/> <code>\providelanguage</code> <hr/>	<code>{\language options} {\language name}</code>
	This is a wrapper command over <code>\babelprovide</code> . The first argument is passed to the optional argument of <code>\babelprovide</code> and the second one to the mandatory argument of the same. For more information, please read <code>babel</code> 's manual. Languages supported by <code>LINGUISTIX-LANGUAGES</code> are loaded with a package with that language's name. If it is absent, the package produces a warning.
<hr/> <code>native numbering</code> <hr/>	<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.
<hr/> <code>strict</code> <hr/>	The 'strict' plug changes the <code>\lngx_counter:n</code> command to the counter of the main language of the document. That way all the counters are printed in the main language.
<hr/> <code>logical</code> <hr/>	This plug changes the meaning of <code>\lngx_counter:n</code> to the <code>\localecounter</code> command provided by <code>babel</code> . 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 <code>TeX</code> '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 <code>TeX</code> 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 <code>TeX</code> .
<hr/> <code>off</code> <hr/>	It is equivalent of the <code>noop</code> plug when the other two are not used at all. It is only required when you want to go back to <code>L^ATeX</code> defaults. E.g., if you have turned strict native numbering in some language and you want it to go back to <code>L^ATeX</code> defaults, you may use this.

This is a small package that provides commands for printing logos of the LINGUIS \mathcal{T} I \mathcal{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` [*$\langle package name \rangle$*]

The logo of the $\langle package name \rangle$ from the LINGUIS \mathcal{T} I \mathcal{X} bundle is printed with this command, e.g., `\lngxlogo[fonts] → LINGUIS \mathcal{T} I \mathcal{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.

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, i.e., encoding, family, shape and 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\ | \fontfamily\ | \fontseries\ | \fontshape\quad
\normalfont
\fontencoding\ | \fontfamily\ | \fontseries\ | \fontshape
\end{document}

```

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

```

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

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

```

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

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

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

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

```

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

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

```

Better? :-)

L^AT_EX₃ interface for programmers

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

LINGUISTIX-BASE

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

<code>\lngx_set_keys:n</code>	<code>⟨<i>keyval list</i>⟩</code>
-------------------------------	-----------------------------------

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 booleans that are used to check if old style numbers, old style one (i.e., ‘1’) and Bourbaki’s empty set symbol (i.e., ‘ \emptyset ’) are asked by the user.

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

`\lngx_set_mono_font:nn` These commands take two arguments, retrieve the values of the data variables if :VV variants are used. These are wrapper commands around the font-setting commands of fontspec and (lua)-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:VV`
`\lngx_set_math_font:nn`
`\lngx_set_math_font:VV`

`\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` 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.
`\lngx_other_mono_font:nee`

LINGUIS \mathcal{T} X-GLOSSING

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

`\lngx_gloss_format:n` $\{\langle gloss \rangle\}$
`\lngx_expansion_format:n` $\{\langle expansion \rangle\}$

This function is controlled by the key `format`. Its argument is the gloss or the expansion itself. According to the definition set in the key, the argument gets printed.

`\lngx_gloss_new:nn` $\{\langle gloss \rangle\}$ $\{\langle expansion \rangle\}$

This function creates a new gloss. It is later equated with the `\newgloss` command.

`\lngx_gloss_list:` This functions prints the list of glosses and is equated with `\listofglosses`.

`lngx_multicols` $\{\langle section\ title\rangle\}$

This environment reads an integer variable, i.e., `\l__lngx_glossary_columns_int`. It is controlled by the `columns` key. If its number is more than one (which, by default *is* more than one), the `multicols` environment is used around the content that comes in between, or else no action is taken. It takes one compulsory argument, i.e., the content of the section title material. This environment should not be used outside this package.

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:VV`
`\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:VV`
`\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`.

`\lngx_set_mono_ipa_font:nn` $\{\langle features\rangle\}$ $\{\langle font\rangle\}$

`\lngx_set_mono_ipa_font:VV`
`\lngx_mono_ipa:`
`lngx_ipa_tt_nfss`

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

`\lngx_ipa:`
`lngx_ipa`

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

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

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

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

`\g_lngx_main_language_tl`

A `tl` that globally stores the main language of the document.

`\g_lngx_languages_clist`

A `clist` that globally stores the languages that are used.

`\lngx_languages:nn` $\{\langle language\ options\rangle\}$ $\{\langle language\ name\rangle\}$

`\lngx_languages:VV` $\langle language\ options\ tl\rangle$ $\langle language\ tl\rangle$

These functions read the V-type argument provided to them and pass it to the `\babelprovide` command for loading languages.

`\lngx_load_languages:n` $\{\langle list\ of\ languages\rangle\}$

This function loads the languages in LINGUIS \mathcal{T} **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} **X**-Logos

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

There are only two \LaTeX 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:  .
--------------------------------	---

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

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

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

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

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

<code>\lngx_if_encoding_p:n</code>	\star <code>{\langle encoding \rangle}</code>
<code>\lngx_if_encoding:nTF</code>	\star <code>{\langle encoding \rangle}{\langle true code \rangle}{\langle false code \rangle}</code>
<code>\lngx_if_meta_family_p:n</code>	\star <code>{\langle meta font family \rangle}</code>
<code>\lngx_if_meta_family:nTF</code>	\star <code>{\langle meta font family \rangle}{\langle true code \rangle}{\langle false code \rangle}</code>
<code>\lngx_if_super_family_p:n</code>	\star <code>{\langle super font family \rangle}</code>
<code>\lngx_if_super_family:nTF</code>	\star <code>{\langle super font family \rangle}{\langle true code \rangle}{\langle false code \rangle}</code>
<code>\lngx_if_series_p:n</code>	\star <code>{\langle series \rangle}</code>
<code>\lngx_if_series:nTF</code>	\star <code>{\langle series \rangle}{\langle true code \rangle}{\langle false code \rangle}</code>
<code>\lngx_if_shape_p:n</code>	\star <code>{\langle shape \rangle}</code>
<code>\lngx_if_shape:nTF</code>	\star <code>{\langle shape \rangle}{\langle true code \rangle}{\langle false code \rangle}</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={\rm NFSS}},sf={\sf NFSS},tt={\tt NFSS}}

```

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

```

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

```

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 {\ID}
\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_{Ti}X

Provide the package with its basic information.

```
1 <*package>
2 \ProvidesExplPackage{linguistix}
3     {2026-05-15}
4     {v1.0}
5     {%
6         The 'LinguisTiX' bundle: Enhanced
7         support for linguistics.%
8     }
```

When one loads LINGUIS_{Ti}X, all the packages of the bundle are loaded automatically. That's the only content of the umbrella package LINGUIS_{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-glossing } {
17     \RequirePackage { linguistix-glossing }
18 }
19 \IfPackageLoadedF { linguistix-ipa } {
20     \RequirePackage { linguistix-ipa }
21 }
22 \IfPackageLoadedF { linguistix-languages } {
23     \RequirePackage { linguistix-languages }
24 }
25 \IfPackageLoadedF { linguistix-leipzig } {
26     \RequirePackage { linguistix-leipzig }
27 }
28 \IfPackageLoadedF { linguistix-logos } {
29     \RequirePackage { linguistix-logos }
30 }
31 \IfPackageLoadedF { linguistix-nfss } {
32     \RequirePackage { linguistix-nfss }
33 }
34 </package>
```


Set the essentials of the package.

```

35 <*base>
36 \ProvidesExplPackage{linguistix-base}
37     {2026-05-15}
38     {v1.0}
39     {%
40         The base package of the ‘LinguisTiX’
41         bundle.%
42     }
```

\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.

```

43
44 \cs_new_protected:Npn \lngx_set_keys:n #1 {
45     \keys_set:nn { lngx_keys } { #1 }
46 }
```

(End of definition for \lngx_set_keys:n. This function is documented on page 19.)

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

```

47
48 \cs_gset_eq:NN \linguistix \lngx_set_keys:n
49 </base>
```

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

The `unicode-math` and `lua-unicode-math` packages define `\gla` command 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 `(lua)-unicode-math-\gla`. This is not relevant to people who are not using `expex`. Thus, the settings are loaded only conditionally.

```

50 \fixpex
51 \ProvidesExplPackage{linguistix-fixpex}
52     {2026-05-15}
53     {v1.0}
54     {%
55         To fix the clash between 'expex' and
56         (lua-unicode-math).%
57     }

```

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

```

58
59 \msg_new:nnn { linguistix-fixpex } { pkg_not_loaded } {
60     This~ package~ is~ a~ first-aid~ for~ resolving~ the\\
61     clash~ between~ '(lua)-unicode-math'~ and~ 'expex'.\\
62     It~ should~ only~ be~ used~ if~ at~ least~ one~ of\\
63     these~ two~ is~ loaded.~ Here~ it~ is~ not~ needed~ as\\
64     '#1'~ is~ not~ loaded.
65 }

```

I first start the hook `begindocument/before`.

```

66
67 \hook_gput_code:nnn { begindocument / before } { . } {

```

The `(lua)-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.

```

68 \IfPackageLoadedTF { expex } {
69     \exp_args:Ne
70     \IfPackageLoadedTF {
71         \sys_if_engine luatex:TF {
72             \IfPackageLoadedF { unicode-math } {
73                 unicode-math
74             } {
75                 lua-unicode-math
76             }
77         } {
78             unicode-math
79         }
80     } {
81         \hook_gput_code:nnn { begindocument / end } { . } {

```

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

```

82         \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).

```

83      \cs_gset_eq:NN \gla    \glw@gla
84    }

```

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

```

85    } {
86      \msg_info:nnn { linguistix-fixpex }
87                  { pkg_not_loaded }
88                  { (lua)-unicode-math }
89    }

```

Similarly, I do it for expex.

```

90    } {
91      \msg_info:nnn { linguistix-fixpex } { pkg_not_loaded } {
92        expex
93      }
94    }
95  }
96 </fixpex>

```

Package essentials first.

```

97 <*font>
98 \ProvidesExplPackage{linguistix-fonts}
99         {2026-05-15}
100        {v1.0}
101        {%
102         The font-assistant package of the
103         'LinguisTiX' bundle.%
104        }

```

Then, I load unicode-math or lua-unicode-math (depending on the engine used), LINGUIS**TiX**-NFSS and LINGUIS**TiX**-BASE (if they are not already loaded).

```

105
106 \IfPackageLoadedF { linguistix-base } {
107   \RequirePackage { linguistix-base }
108 }
109
110 \sys_if_engine luatex:TF {
111   \IfPackageLoadedF { unicode-math } {
112     \IfPackageLoadedF { lua-unicode-math } {
113       \RequirePackage { fontspec, lua-unicode-math }
114     }
115   }
116 } {
117   \IfPackageLoadedF { unicode-math } {
118     \RequirePackage { unicode-math }
119   }
120 }
121
122 \IfPackageLoadedF { linguistix-fixpex } {
123   \RequirePackage { linguistix-fixpex }
124 }

```

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

```

125
126 \NewCommandCopy \ogLaTeX \LaTeX
127
128 \RenewDocumentCommand \LaTeX { } {%
129   L\kern-.81ex\relax
130   \raisebox{.6ex}{\textsc{a}}\kern-.23ex\relax
131   \hbox{T}\kern-.4ex\relax
132   \raisebox{-.5ex}{E}\kern-.3ex\relax
133   X%
134 }

```

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

```

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

```

I used to set the `old style numbers` key to true, but starting from version 0.9c, I have made this selection conditional. Only when the language package is not loaded (till the end of the preamble), I do that. Otherwise respective language packages set it. If the user explicitly decides to set it to some value, then the package shouldn't change the value with the hook. In order to check that, we develop an internal boolean.

```

135
136 \bool_new:N \g_lngx_old_style_bool
137 \bool_new:N \g__lngx_old_style_user_choice_bool
138
139 \keys_define:nn { lngx_keys } {
140   old~ style~ numbers
141   .choices:nn          = { true,false } {
142     \bool_gset_true:N \g__lngx_old_style_user_choice_bool
143     \use:c { bool_gset_ \l_keys_choice_str :N }
144     \g_lngx_old_style_bool
145   },
146   old~ style~ numbers
147   .default:n           = { true },
148   old~ style~ one
149   .bool_gset:N          = \g_lngx_old_style_one_bool,
150   bourbaki's~ empty~ set
151   .bool_gset:N          = \g_lngx_bourbaki_bool
152 }

```

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

```

\g_lngx_text_main_fonts_prop
\g__lngx_text_main_font_features_tl
  text upright
text upright features
  text bold
  text bold 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` lists that save 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>.)

```

153
154 \prop_gclear_new:N \g__lngx_text_main_fonts_prop
155 \tl_gclear_new:N \g__lngx_text_main_font_features_tl
156
157 \clist_map_inline:nn {
158   upright,
159   bold,
160   italic,
161   bold~ italic,
162   slanted,
163   bold~ slanted,
164   swash,
165   bold~ swash,
166   small~ caps
167 } {

```

All the keys listed here use a global token list to have an appending list of features. The variables are declared here.

```

168 \group_begin:
169 \str_clear:N \l_tmpa_str
170 \str_set:Nn \l_tmpa_str { #1 }
171 \str_replace_all:Nnn \l_tmpa_str { ~ } { _ }

```

```

172 \tl_gclear_new:c {
173   g__lngx_text_main_features_ \l_tmpa_str _tl
174 }
175 \group_end:

```

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.

```

176 \keys_define:nn { lngx_keys } {
177   text~ #1
178   .code:n          = {

```

I start a group first. Then clear and set a temporary string variable. I make the text of the key titlecased as required by `fontspec` and remove the spaces. Lastly, the word `Font` is appended. So, bold italic becomes `BoldItalicFont`.

```

179   \group_begin:
180   \str_clear:N \l_tmpa_str
181   \str_set:Ne \l_tmpa_str {
182     \text_titlecase_all:n { #1 }
183     Font
184   }
185   \str_replace_all:Nnn \l_tmpa_str { ~ } { }

```

The string is used inside the relevant `prop`-key and group is ended.

```

186   \prop_gput:Nne \g__lngx_text_main_fonts_prop
187               { text~ #1 }
188               { \str_use:N \l_tmpa_str = { ##1 } }
189   \group_end:
190 },

```

Same is repeated for features, but we add the value of the keys to a continuously appending token list first and then expand it in the `prop` variable.

```

191   text~ #1~ features
192   .code:n          = {
193     \group_begin:
194     \str_clear:N \l_tmpa_str
195     \str_clear:N \l_tmpb_str
196     \str_set:Ne \l_tmpa_str {
197       \text_titlecase_all:n { #1 }
198       Features
199     }
200     \str_set:Nn \l_tmpb_str { #1 }
201     \str_replace_all:Nnn \l_tmpa_str { ~ } { }
202     \str_replace_all:Nnn \l_tmpb_str { ~ } { _ }
203     \tl_gput_right:ce {
204       g__lngx_text_main_features_ \l_tmpb_str _tl
205     } {
206       \tl_if_empty:cF {
207         g__lngx_text_main_features_ \l_tmpb_str _tl
208       } { , }
209       \exp_not:n { ##1 }
210     }
211     \prop_gput:Nne \g__lngx_text_main_fonts_prop

```

```

212         { text~ #1~ features } {
213         \str_use:N \l_tmpa_str = {
214             \exp_not:v {
215                 g__lngx_text_main_features_ \l_tmpb_str _tl
216             }
217         }
218     }
219     \group_end:
220 }
221 }
222 }

```

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

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

```

223
224 \tl_gclear_new:N \g__lngx_text_main_features_extra_tl
225
226 \keys_define:nn { lngx_keys } {
227     text~ main~ extra~ features
228         .code:n = {
229             \tl_gput_right:N \g__lngx_text_main_features_extra_tl {
230                 \tl_if_empty:NF \g__lngx_text_main_features_extra_tl {
231                     ,
232                 }
233             \exp_not:n { #1 }
234         }
235     \prop_gput:Nne \g__lngx_text_main_fonts_prop
236         { text~ main~ extra~ features } {
237         \exp_not:V \g__lngx_text_main_features_extra_tl
238     }
239 }
240 }

```

(End of definition for `text main extra features`. This function is documented on page [I3.](#))

```

\g__lngx_text_sans_fonts_prop
\g__lngx_text_sans_font_features_tl
\g__lngx_text_mono_fonts_prop
\g__lngx_text_mono_font_features_tl
text sans extra features
    text sans upright
text sans upright features
    text sans bold
    text sans bold 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 extra features
    text mono upright
text mono upright features
    text mono bold
    text mono bold 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.

```

241
242 \prop_gclear_new:N \g__lngx_text_sans_fonts_prop
243 \tl_gclear_new:N \g__lngx_text_sans_font_features_tl
244
245 \prop_gclear_new:N \g__lngx_text_mono_fonts_prop
246 \tl_gclear_new:N \g__lngx_text_mono_font_features_tl
247
248 \clist_map_inline:nn {
249     sans,
250     mono
251 } {
252     \tl_gclear_new:c {
253         g__lngx_text_ #1 _features_extra_tl
254     }
255     \clist_map_inline:nn {
256         upright,
257         bold,
258         italic,
259         bold~ italic,
260         slanted,
261         bold~ slanted,
262         swash,
263         bold~ swash,
264         small~ caps
265     } {
266         \group_begin:
267         \str_clear:N \l_tmpa_str
268         \str_set:Nn \l_tmpa_str { ##1 }
269         \str_replace_all:Nnn \l_tmpa_str { ~ } { _ }
270         \tl_gclear_new:c {
271             g__lngx_text_ #1 _features_ \l_tmpa_str _tl
272         }
273         \group_end:
274         \keys_define:nn { lngx_keys } {
275             text~ #1~ ##1
276             .code:n = {
277                 \group_begin:
278                 \str_clear:N \l_tmpa_str
279                 \str_set:Ne \l_tmpa_str {
280                     \text_titlecase_all:n { ##1 }
281                     Font
282                 }
283                 \str_replace_all:Nnn \l_tmpa_str { ~ } { }
284                 \prop_gput:cne { g__lngx_text_ #1 _fonts_prop }
285                     { text~ #1~ ##1 } {
286                     \str_use:N \l_tmpa_str = {
287                         \exp_not:n { #####1 }
288                     }
289                 }
290                 \group_end:
291             },
292             text~ #1~ ##1~ features

```



```

293 .code:n = {
294   \group_begin:
295   \str_clear:N \l_tmpa_str
296   \str_clear:N \l_tmpb_str
297   \str_set:N \l_tmpa_str {
298     \text_titlecase_all:n { ##1 }
299     Features
300   }
301   \str_set:Nn \l_tmpb_str { ##1 }
302   \str_replace_all:Nnn \l_tmpa_str { ~ } { }
303   \str_replace_all:Nnn \l_tmpb_str { ~ } { _ }
304   \tl_gput_right:ce {
305     g__lngx_text_ #1 _features_ \l_tmpb_str _tl
306   } {
307     \tl_if_empty:cF {
308       g__lngx_text_ #1 _features_ \l_tmpb_str _tl
309     } { , }
310     \exp_not:n { ####1 }
311   }
312   \prop_gput:cne { g__lngx_text_ #1 _fonts_prop }
313     { text~ #1~ ##1~ features } {
314     \str_use:N \l_tmpa_str = {
315       \exp_not:v {
316         g__lngx_text_ #1 _features_ \l_tmpb_str _tl
317       }
318     }
319   }
320   \group_end:
321 }
322 }
323 }
324 \keys_define:nn { lngx_keys } {
325   text~ #1~ extra~ features
326   .code:n = {
327     \tl_gput_right:ce {
328       g__lngx_text_ #1 _features_extra_tl
329     } {
330       \tl_if_empty:cF {
331         g__lngx_text_ #1 _features_extra_tl
332       } { , }
333       \exp_not:n { ##1 }
334     }
335     \prop_gput:cne { g__lngx_text_ #1 _fonts_prop }
336       { text~ #1~ extra~ features } {
337       \exp_not:v { g__lngx_text_ #1 _features_extra_tl }
338     }
339   }
340 }
341 }

```

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

```

\g__lngx_text_main_font_tl
\g__lngx_text_sans_font_tl
\g__lngx_text_mono_font_tl

```

```

text main font
text sans font
text mono font

```

These keys add the parameter that sets the main font for text. They set an internal token list which is retrieved later by font setting command.

```

342
343 \clist_map_inline:nn {
344     main,
345     sans,
346     mono
347 } {
348     \keys_define:nn { lngx_keys } {
349         text~ #1~ font
350         .tl_gset:c          = { g__lngx_text_ #1 _font_tl }
351     }
352 }

```

(End of definition for `\g__lngx_text_main_font_tl` and others. These functions are documented on page 11.)

```

\g__lngx_math_fonts_prop
\g__lngx_math_features_tl
\g__lngx_math_bold_fonts_prop
\g__lngx_math_bold_features_tl
    math
    math features
    math bold
    math bold features

```

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

```

353
354 \prop_gclear_new:N \g__lngx_math_fonts_prop
355 \tl_gclear_new:N \g__lngx_math_features_tl
356
357 \prop_gclear_new:N \g__lngx_math_bold_fonts_prop
358 \tl_gclear_new:N \g__lngx_math_bold_features_tl
359
360 \keys_define:nn { lngx_keys } {
361     math
362     .tl_gset:N          = \g__lngx_math_font_tl,
363     math~ bold
364     .tl_gset:N          = \g__lngx_math_bold_font_tl,
365     math~ features
366     .prop_gput:N        = \g__lngx_math_fonts_prop,
367     math~ bold~ features
368     .prop_gput:N        = \g__lngx_math_bold_fonts_prop
369 }

```

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

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.

```

370
371 \keys_define:nn { lngx_keys } {
372     newcm
373     .meta:n          = {
374         text~ main~ font      = { NewCM10-Book.otf },
375         text~ sans~ font     = { NewCMSans10-Book.otf },
376         text~ mono~ font     = { NewCMMono10-Book.otf },
377         math                 = { NewCMMath-Book.otf },
378         math~ bold          = { NewCMMath-Bold.otf }
379     }
380 }

```

(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.

```

381
382 \keys_define:nn { lngx_keys } {
383     newcm~ sans

```

```

384 .meta:n          = {
385   text~ main~ font = { NewCMSans10-Book.otf },
386   text~ sans~ font = { NewCMSans10-Book.otf },
387   text~ mono~ font = { NewCMMono10-Book.otf },
388   math            = { NewCMSansMath-Regular.otf },
389   math~ bold      = { NewCMSansMath-Regular.otf }
390 }
391 }

```

(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.

```

392
393 \keys_define:nn { lngx_keys } {
394   newcm~ mono
395   .meta:n          = {
396     text~ main~ font = { NewCMMono10-Book.otf },
397     text~ sans~ font = { NewCMSans10-Book.otf },
398     text~ mono~ font = { NewCMMono10-Book.otf },
399     math            = { NewCMSansMath-Regular.otf },
400     math~ bold      = { NewCMSansMath-Regular.otf }
401   }
402 }

```

(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.

```

403
404 \keys_define:nn { lngx_keys } {
405   newcm~ regular
406   .meta:n          = {
407     text~ main~ font = { NewCM10-Regular.otf },
408     text~ sans~ font = { NewCMSans10-Regular.otf },
409     text~ mono~ font = { NewCMMono10-Regular.otf },
410     math            = { NewCMMath-Regular.otf },
411     math~ bold      = { NewCMMath-Bold.otf }
412   }
413 }

```

(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.

```

414
415 \keys_define:nn { lngx_keys } {
416   newcm~ regular~ sans
417   .meta:n          = {
418     text~ main~ font = { NewCMSans10-Regular.otf },
419     text~ sans~ font = { NewCMSans10-Regular.otf },
420     text~ mono~ font = { NewCMMono10-Regular.otf },
421     math            = { NewCMMath-Regular.otf },
422     math~ bold      = { NewCMMath-Bold.otf }
423   }
424 }

```

(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.

```

425
426 \keys_define:nn { lngx_keys } {
427   newcm~ regular~ mono
428   .meta:n = {
429     text~ main~ font = { NewCMMono10-Regular.otf },
430     text~ sans~ font = { NewCMSans10-Regular.otf },
431     text~ mono~ font = { NewCMMono10-Regular.otf },
432     math = { NewCMMath-Regular.otf },
433     math~ bold = { NewCMMath-Bold.otf },
434   }
435 }
```

(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.

```

436
437 \lngx_set_keys:n {
438   bourbaki's~ empty~ set,
```

Then we load the `old style numbers` boolean.

```

439   newcm
440 }
```

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

```

441
442 \IfPackageLoadedF { linguistix-languages } {
443   \cs_new_protected:Npn \lngx_set_main_font:nn #1#2 {
444     \setmainfont [ #1 ] { #2 }
445   }
446   \cs_new_protected:Npn \lngx_set_sans_font:nn #1#2 {
447     \setsansfont [ #1 ] { #2 }
448   }
449   \cs_new_protected:Npn \lngx_set_mono_font:nn #1#2 {
450     \setmonofont [ #1 ] { #2 }
451   }
452 }
```

A wrapper command is provided for loading math fonts.

```

453
454 \cs_new_protected:Npn \lngx_set_math_font:nn #1#2 {
455   \setmathfont [ #1 ] { #2 }
456 }
457
458 \cs_new_protected:Npn \lngx_set_math_bold_font:nn #1#2 {
459   \IfPackageLoadedT { lua-unicode-math } {
460     \DeclareMathVersion { bold }
461   }
462   \setmathfont [
```

```

463     #1,
464     version                = { bold }
465 ] { #2 }
466 }

```

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

```

467
468 \cs_generate_variant:Nn \lngx_set_main_font:nn { VV }
469 \cs_generate_variant:Nn \lngx_set_sans_font:nn { VV }
470 \cs_generate_variant:Nn \lngx_set_mono_font:nn { VV }
471 \cs_generate_variant:Nn \lngx_set_math_font:nn { VV }
472 \cs_generate_variant:Nn \lngx_set_math_bold_font:nn { VV }

```

(End of definition for `\lngx_set_main_font:nn` and others. These functions are documented on page 20.)

```

\__lngx_build_main_font_features:
\__lngx_build_sans_font_features:
\__lngx_build_mono_font_features:
\__lngx_build_math_font_features:
\__lngx_build_bold_math_font_features:
\g__lngx_text_main_font_features_tl
\g__lngx_text_sans_font_features_tl
\g__lngx_text_mono_font_features_tl
\g__lngx_math_font_features_tl
\g__lngx_bold_math_font_features_tl

```

These are some internal functions that basically iterate on the `prop` list items and each of them is put to the right of the respective token list. This way only the functions that are added by the user are exported to the font setting command.

```

473
474 \clist_map_inline:nn {
475     main,
476     sans,
477     mono
478 } {
479     \cs_new_protected:cpn {
480         __lngx_build_#1_font_features:
481     } {
482         \prop_map_inline:cn { g__lngx_text_#1_fonts_prop } {
483             \tl_gput_right:cn {
484                 g__lngx_text_#1_font_features_tl
485             } { ####2, }
486         }
487     }
488 }
489
490 \cs_new_protected:Npn \__lngx_build_math_features: {
491     \prop_map_inline:Nn \g__lngx_math_fonts_prop {
492         \tl_gput_right:Nn \g__lngx_math_features_tl {
493             { ##2 }
494         }
495     }
496 }
497
498 \cs_new_protected:Npn \__lngx_build_math_bold_features: {
499     \prop_map_inline:Nn \g__lngx_math_bold_fonts_prop {
500         \tl_gput_right:Nn \g__lngx_math_bold_features_tl {
501             { ##2 }
502         }
503     }
504 }

```

(End of definition for `__lngx_build_main_font_features:` and others.)

Now I start the pre-begindocument hook.

```

505
506 \hook_gput_code:nnn { begindocument / before } { . } {

```

Starting from v0.9c, I have decided to set the `old style numbers` key conditionally. The reason for this change is explained in the documentation of `old style numbers` key. Here, I load this key only if `LINGUISTIX-LANGUAGES` is not loaded. In that case we can assume that the author has no intention to write a multilingual document. Thus, it can be assumed that the document is supposed to be in English and there we can make this mandatory. Also, if the user has set this option themselves, that means the package shouldn't tinker with it. So we check it with the internal boolean that was set along with the key.

```

507 \IfPackageLoadedF { linguistix-languages } {
508   \bool_if:NF \g__lngx_old_style_user_choice_bool {
509     \lngx_set_keys:n { old~ style~ numbers }
510   }
511 }

```

If the boolean for old style numbers is true, I set the `Numbers` key to `OldStyle`. Similarly, if the NewCM-specific old one is requested, I turn the character-variant on.

```

512 \lngx_set_keys:n {
513   text~ main~ extra~
514   features = {
515     \bool_if:NT \g__lngx_old_style_bool {
516       Numbers = { OldStyle },
517     \bool_if:NT \g__lngx_old_style_one_bool {
518       CharacterVariant = { 6 }
519     }
520   }
521 },
522   text~ sans~ extra~
523   features = {
524     \bool_if:NT \g__lngx_old_style_bool {
525       Numbers = { OldStyle },
526     \bool_if:NT \g__lngx_old_style_one_bool {
527       CharacterVariant = { 6 }
528     }
529   }
530 }
531 }

```

All the font features are built using the internal functions and then fonts are set.

```

532 \__lngx_build_main_font_features:
533 \lngx_set_main_font:VV
534   \g__lngx_text_main_font_features_tl
535   \g__lngx_text_main_font_tl
536 \__lngx_build_sans_font_features:
537 \lngx_set_sans_font:VV
538   \g__lngx_text_sans_font_features_tl
539   \g__lngx_text_sans_font_tl
540 \__lngx_build_mono_font_features:
541 \lngx_set_mono_font:VV
542   \g__lngx_text_mono_font_features_tl
543   \g__lngx_text_mono_font_tl
544 \__lngx_build_math_features:
545 \lngx_set_math_font:VV \g__lngx_math_features_tl

```

```

546             \g__lngx_math_font_tl
547 \IfPackageLoadedT { unicode-math } {
548   \__lngx_build_math_bold_features:
549   \lngx_set_math_bold_font:VV
550   \g__lngx_math_bold_features_tl
551   \g__lngx_math_bold_font_tl
552 }
553 }
554 \font>

```

```

555 <*glossing>
556 \ProvidesExplPackage{linguistix-glossing}
557             {2026-05-15}
558             {v1.0}
559             {%
560             Accessible glossing with Linguistix%
561             }

```

In order to print the multi-column glossary, I load the `\multicol` package.

```

562
563 \IfPackageLoadedF { multicol } {
564   \RequirePackage { multicol }
565 }

```

Then I declare some variables that will be used for generating the glossing-auxiliary.

```

566
567 \bool_new:N      \l_lngx_expansion_bool
568 \tl_clear_new:N  \l_lngx_gloss_separator_tl
569 \tl_clear_new:N  \l_lngx_expansion_separator_tl
570 \tl_clear_new:N  \l_lngx_glossary_separator_tl
571 \dim_zero_new:N  \l_lngx_i_have_dim
572 \dim_zero_new:N  \l_lngx_i_need_dim
573 \dim_zero_new:N  \l_lngx_remain_dim
574 \dim_zero_new:N  \l_lngx_i_hack_dim
575 \int_gzero_new:N \g__lngx_page_ref_int
576 \str_clear_new:N \l_lngx_gls_language_str
577 \str_clear_new:N \l__lngx_gls_sorting_order_str
578 \str_clear_new:N \l__lngx_gls_expansion_case_str
579 \str_clear_new:N \l__lngx_glossary_style_str
580 \str_clear_new:N \l__lngx_separator_str
581 \seq_gclear_new:N \g__lngx_gls_use_order_seq
582
583 \str_set:Nn \l__lngx_separator_str { gloss }

```

Glossaries are hyperlinked with complex and cryptic labels. Some readers read the labels loudly when using assistive technology. In order to dodge that, I add the text to the Contents key. It uses Ulrike's ideas: tex.stackexchange.com/a/758083/174620.

```

584
585 \IfPDFManagementActiveT {
586   \socket_if_exist:nT { hyp / link / GoTo / Contents } {
587     \socket_new_plug:nnn { hyp / link / GoTo / Contents }
588       { text } {
589         \pdfstringdef \__lngx_tmp_text: { #2 }
590         \pdfannot_dict_put:nne { link / GoTo } { Contents } {
591           ( \__lngx_tmp_text: )
592         }
593       }
594   }
595 }

```

After these initial declarations, I move to the socket that controls the description of the gloss. The socket itself has no arguments.

```

596
597 \socket_new:nn { lngx / description / gloss } { 0 }

```


`__lngx_gloss_description:` When the socket is assigned the `on` plug, it defines the expandable internal command for glossing description. It is then used inside the tagging socket. The same command is made inactive when the socket is assigned the `off` plug. By default the `off` plug is assigned (this is experimental and may change after reviews from the blind people). The socket is activated by using it.

```

598
599 \socket_new_plug:nnn { lngx / description / gloss } { on } {
600   \cs_set:Npn \__lngx_gloss_description: { Gloss~ }
601 }
602
603 \socket_new_plug:nnn { lngx / description / gloss }
604   { off } {
605   \cs_set_eq:NN \__lngx_gloss_description: \prg_do_nothing:
606 }
607
608 \socket_assign_plug:nn { lngx / description / gloss }
609   { off }
610
611 \socket_use:n { lngx / description / gloss }

```

(End of definition for `__lngx_gloss_description:`.)

Then I declare the tagging socket for glossing which takes two arguments. It should follow the default tagging which is why I use the `default` plug (which is the only plug the package does and will offer). The code is based on suggestions by Ulrike Fischer (github.com/latex3/tagging-project/discussions/975). The `E` tag is used for ‘expansion’ which more or less suits the nature of glosses. So it is used here. The command `__lngx_gloss_description:` is controlled by the socket and is expandable.

```

612
613 \NewTaggingSocket { lngx / gloss } { 2 }
614
615 \NewTaggingSocketPlug { lngx / gloss } { default } {
616   \mode_leave_vertical:
617   \tag_mc_end:
618   \exp_args:Ne
619   \tag_struct_begin:n {
620     tag                = { Span },
621     E                  = {
622       \__lngx_gloss_description: #2
623     }
624   }
625   \tag_mc_begin:n {
626     tag                = { Span }
627   }

```

The argument is printed with the package-controlled formatting command. First I check if the `hyperref` package is loaded. If it is loaded, the link colour is changed to the one stored in the variable `\g_lngx_gloss_link_color_str` (black, by default).

```

628 \IfPackageLoadedTF { hyperref } {
629   \group_begin:
630   \str_clear:N \l_tmpa_str
631   \str_set:Nn \l_tmpa_str { #1 }
632   \exp_args:Ne \hypersetup {
633     linkcolor      = {

```

```

634         \exp_not:V \g__lngx_gloss_link_color_str
635     }
636 }

```

The socket for adding text into the Contents directory is used here.

```

637 \IfPDFManagementActiveT {
638     \socket_if_exist:nT { hyp / link / GoTo / Contents } {
639         \socket_assign_plug:nn {
640             hyp / link / GoTo / Contents
641         } { text }
642     }
643 }
644 \lngx_gloss_format:n {
645     \hyperlink { lngx_ #1 _glossary } { #1 }
646 }
647 \group_end:
648 } {

```

If `hyperref` is not loaded, the text is simply printed with the formatting command.

```

649     \lngx_gloss_format:n { #1 }
650 }
651 \tag_mc_end:
652 \tag_struct_end:
653 \tag_mc_begin:n { }
654 }

```

I assign the default tagging plug to the socket I just defined.

```

655
656 \AssignTaggingSocketPlug { lngx / gloss } { default }

```

format Now I define the key for adjusting the formatting of the glosses. It controls several keys contained in a separate set. In short, this key will take another keys as arguments.

```

657
658 \keys_define:nn { lngx_glossing } {
659     format
660     .meta:nn          = { lngx / gloss / format } { #1 },

```

(End of definition for `format`. This function is documented on page 9.)

link color This option sets the colour used for glossing links. It is set to `black` by default.

```

\g__lngx_gloss_link_color_str 661 link~ color
662 .str_gset:N                  = \g__lngx_gloss_link_color_str,
663 link~ color
664 .initial:n                   = { black },

```

(End of definition for `link color` and `\g__lngx_gloss_link_color_str`. This function is documented on page 9.)

sort Glosses can be sorted alphabetically or as they are used. The choice key for that is as follows. By default glosses are sorted alphabetically.

```

\l__lngx_gls_sorting_order_str 665 sort
666 .choices:nn                  = { alphabetical, use } {
667     \str_set_eq:NN \l__lngx_gls_sorting_order_str
668                     \l_keys_choice_str
669 },
670 sort
671 .initial:n                   = { alphabetical },

```

(End of definition for `sort` and `\l__lngx_gls_sorting_order_str`. This function is documented on page 9.)

expansion case `\l__lngx_gls_expansion_case_str` The expansion can be printed in lower case, title case (with the first letter capitalised for all the words) or title case (with the first letter capitalised only for the first word). The default is lower case.

```

672 expansion~ case
673 .choices:nn          = {
674   lowercase, title~ case~ all, title~ case~ first
675 } {
676   \str_set_eq:NN \l__lngx_gls_expansion_case_str
677                 \l_keys_choice_str
678 },
679 expansion~ case
680 .initial:n          = { lowercase },

```

(End of definition for `expansion case` and `\l__lngx_gls_expansion_case_str`. This function is documented on page 9.)

style `\l__lngx_glossary_style_str` The glossary can be printed in two styles given below. The default is `block`.

```

681 style
682 .choices:nn          = { block, inline } {
683   \str_set_eq:NN \l__lngx_glossary_style_str
684                 \l_keys_choice_str
685 },
686 style
687 .initial:n          = { block },

```

(End of definition for `style` and `\l__lngx_glossary_style_str`. This function is documented on page 9.)

columns `\l__lngx_glossary_columns_int` There is an option to change the number of columns used for printing the glossary. It is controlled here. Default is 2.

```

688 columns
689 .int_set:N           = \l__lngx_glossary_columns_int,
690 columns
691 .initial:n           = { 2 },

```

(End of definition for `columns` and `\l__lngx_glossary_columns_int`. This function is documented on page 10.)

page numbers `\l__lngx_glosses_page_number_bool` Page numbers can be turned off with the following boolean. By default, they are active.

```

692 page~ numbers
693 .bool_set:N          =
694   \l__lngx_glosses_page_number_bool,
695 page~ numbers
696 .initial:n           = { true },

```

(End of definition for `page numbers` and `\l__lngx_glosses_page_number_bool`. This function is documented on page 10.)

sectioning `\l__lngx_gls_sectioning_str` The section used for printing the glossary title is controlled by the following command. By default, I use `\section` for printing the title.

```

697 sectioning
698 .str_set:N           = \l__lngx_gls_sectioning_str,
699 sectioning
700 .initial:n           = { section },

```

(End of definition for sectioning and \l__lngx_gls_sectioning_str. This function is documented on page 10.)

section number This controls if the sectioning level should be numbered or unnumbered. The default is false.
\l__lngx_gls_section_number_bool

```

701 section~ number
702 .bool_set:N          = \l__lngx_gls_section_number_bool,
703 section~ number
704 .initial:n           = { false },

```

(End of definition for section number and \l__lngx_gls_section_number_bool. This function is documented on page 10.)

no bold The no bold key is defined as an inverse boolean. By default the key is set to false (resulting in the controlled boolean being true).
\l__lngx_gls_bold_bool

```

705 no~ bold
706 .bool_set_inverse:N   = \l__lngx_gls_bold_bool,
707 no~ bold
708 .initial:n            = { false },

```

(End of definition for no bold and \l__lngx_gls_bold_bool. This function is documented on page 10.)

separator The separator between the glosses is controlled using this key. It controls the separator for inline glosses, expansion of glosses as well as glosses seen in the glossary. Each of these functions set a string variable which is expanded when this key is used. The default value of the string variable is gloss and the default value for this key is ,~, which means by default the separator between glosses is a comma followed by a space.
\l__lngx_separator_tl

```

709 separator
710 .code:n              = {
711   \tl_set:cn {
712     l__lngx_
713     \str_use:N \l__lngx_separator_str
714     _separator_tl
715   } { #1 }
716 },
717 separator
718 .initial:n            = { ,~ },

```

(End of definition for separator and \l__lngx_separator_tl. This function is documented on page 10.)

entry separator The separator between glossary entries is controlled using this key. The default is a \par token.
\l__lngx_entry_separator_tl

```

719 entry~ separator
720 .tl_set:N            = \l__lngx_entry_separator_tl,
721 entry~ separator
722 .initial:n            = { \par }
723 }

```

(End of definition for entry separator and \l__lngx_entry_separator_tl. This function is documented on page 10.)

Sometimes language-specific settings are needed. I define the language string variable with the information retrieved from the lang key of the PDF.

```

724
725 \IfPDFManagementActiveT {

```

```

726 \str_set:Ne \l_lngx_gls_language_str {
727   \GetDocumentProperties { document / lang }
728 }
729 }

```

gloss The formatting of glosses is defined here. By default they are printed in small caps.

```

\lngx_gloss_format:n
730
731 \keys_define:nn { lngx / gloss / format } {
732   gloss
733   .cs_gset_protected:Np = \lngx_gloss_format:n #1,
734   gloss
735   .initial:n           = { \textsc { #1 } },

```

(End of definition for *gloss* and *\lngx_gloss_format:n*. These functions are documented on page 9.)

expansion The formatting of expansions is defined here. There is no change in the printing in the defaults.

```

\lngx_expansion_format:n
736   expansion
737   .cs_gset_protected:Np = \lngx_expansion_format:n #1,
738   expansion
739   .initial:n           = { #1 }
740 }

```

(End of definition for *expansion* and *\lngx_expansion_format:n*. These functions are documented on page 9.)

\setupglossing A wrapper around these options is provided.

```

741
742 \NewDocumentCommand \setupglossing { m } {
743   \keys_set:nn { lngx_glossing } { #1 }
744 }

```

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

\newgloss A function that creates new glosses starts here. It takes 2 arguments.

```

\lngx_gloss_new:nn
745
746 \cs_new_protected:Npn \lngx_gloss_new:nn #1#2 {

```

First and foremost, the string received as the first argument should change its case to lowercase. It is done by `\str_lowercase:n`. I will use a temporary string variable for storing the converted value. This needs to be done locally so I start a group and clear the local `str` variable.

```

747   \group_begin:
748   \str_clear:N \l_tmpa_str
749   \str_set:Ne \l_tmpa_str { \str_lowercase:n { #1 } }

```

Every gloss has its expansion stored in a token list associated to it. The token list is declared here and it is set to the expansion (i.e., #2).

```

750   \tl_gclear_new:c {
751     g_lngx_ \str_use:N \l_tmpa_str _expansion_tl
752   }
753   \seq_gclear_new:c {
754     g_lngx_ \str_use:N \l_tmpa_str _pages_seq
755   }
756   \tl_gset:cn {
757     g_lngx_ \str_use:N \l_tmpa_str _expansion_tl
758   } { #2 }

```

Whenever a gloss is defined, an internal protected command is defined. It doesn't take any argument.

```

759 \cs_new_protected:cpn {
760   __lngx_gloss_ \str_use:N \l_tmpa_str :
761 } {

```

The arguments are passed to the tagging socket. Since the tagging socket doesn't expand everything, an exhaustive expansion is performed with the help of `\exp_args:Nee`. This is done only if the `\DocumentMetadata` command is used.

```

762 \IfDocumentMetadataTF {
763   \exp_args:Nee \UseTaggingSocket
764     { lngx / gloss }
765     { \str_use:N \l_tmpa_str }
766     { #2 }
767 } {
768   \IfPackageLoadedTF { hyperref } {
769     \group_begin:
770     \exp_args:Ne \hypersetup {
771       linkcolor = {
772         \exp_not:V \g__lngx_gloss_link_color_str
773       }
774     }
775     \IfPDFManagementActiveT {
776       \socket_if_exist:nT {
777         hyp / link / GoTo / Contents
778       } {
779         \socket_assign_plug:nn {
780           hyp / link / GoTo / Contents
781         } { text }
782       }
783     }
784     \lngx_gloss_format:n {
785       \hyperlink { lngx_ #1 _glossary } { #1 }
786     }
787     \group_end:
788   } {
789     \lngx_gloss_format:n { #1 }
790   }
791 }

```

I use `\label-\ref` mechanism for saving the page numbers of the glosses. An internal integer called `\g__lngx_page_ref_int` is used to generate unique numbers. The kernel provides `\seq_remove_duplicates:N`, but as it iterates on each and every item, it is slow. The duplicates can be avoided if the items are added to the sequence conditionally and only when they don't exist already in the sequence. This way duplicates are not generated at all. This method is used for adding to the sequences that respectively store the page numbers of glosses and the order in which they were used. Imagine if a gloss is used twice on a page, it doesn't make sense to add the same page number twice. Similarly, if a gloss is used, it is added to the sequence of used glosses. It doesn't make sense to add it 10 times again and removing the 9 duplicates later.

```

792 \int_gincr:N \g__lngx_page_ref_int
793 \exp_args:Ne
794 \label { lngx_gloss_ \int_use:N \g__lngx_page_ref_int }
795 \cs_if_exist:cT {

```

```

796     r @ lngx_gloss_ \int_use:N \g__lngx_page_ref_int
797 } {
798   \group_begin:
799   \tl_clear:N \l_tmpa_tl
800   \tl_set:N \l_tmpa_tl {
801     \exp_not:N \use_ii:nnnnn
802     \use:c {
803       r @ lngx_gloss_ \int_use:N \g__lngx_page_ref_int
804     }
805   }
806   \seq_if_in:cVF {
807     g_lngx_ \str_use:N \l_tmpa_str _pages_seq
808   } \l_tmpa_tl {
809     \seq_gput_right:ce {
810       g_lngx_ \str_use:N \l_tmpa_str _pages_seq
811     } {
812       \exp_not:N \use_ii:nnnnn
813       \use:c {
814         r @ lngx_gloss_ \int_use:N \g__lngx_page_ref_int
815       }
816     }
817   }
818   \group_end:
819 }
820 \seq_if_in:N \g__lngx_gls_use_order_seq {
821   \str_use:N \l_tmpa_str
822 } {
823   \seq_gput_right:Ne \g__lngx_gls_use_order_seq
824     { \str_use:N \l_tmpa_str }
825 }
826 }
827 \group_end:
828 }
829
830 \cs_gset_eq:NN \newgloss \lngx_gloss_new:nn

```

(End of definition for `\newgloss` and `\lngx_gloss_new:nn`. These functions are documented on page 8.)

\renewgloss Implementing the `\renewgloss` command is actually quite easy. The definition of `\lngx_gloss_new:nn` uses only a single command that errors if the control sequence is already defined, i.e., `\cs_new_protected:cpn`. In order to renew a gloss, simply undefining the existing command declared with `\lngx_gloss_new:nn` suffices. Later the arguments are passed to the same command again. No L^AT_EX₃ equivalent for this is provided.

```

831
832 \NewDocumentCommand \renewgloss { m m } {
833   \cs_undefine:c { __lngx_gloss_ #1 : }
834   \lngx_gloss_new:nn { #1 } { #2 }
835 }

```

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

\glx The command to use a gloss takes three arguments where the first is an optional asterisk. If it is used, the expansion of the gloss is printed without any special tags, just as plain text. Otherwise the internal command for printing the gloss is used with the third argument.

The third argument is a clist. Any number of glosses can be added to the list. The action is then repeated on each and every item of the list. The second argument is a list of options for customising the output. Everything is computed locally so that for the settings don't leak. I perform the action on the first item as desired, then the same is applied to the remaining items with a preceding separator. So that all the items are separated properly.

```

836
837 \NewDocumentCommand \glx { s O{ } m } {
838   \group_begin:
839   \IfBooleanT { #1 } {
840     \bool_set_true:N \l_lngx_expansion_bool
841     \str_set:Nn \l__lngx_separator_str { expansion }
842     \keys_set:nn { lngx_glossing } {
843       separator = { , \c_space_tl }
844     }
845   }
846   \keys_set:nn { lngx_glossing } { #2 }
847   \tl_clear:N \l_tmpa_tl
848   \seq_clear:N \l_tmpa_seq
849   \seq_set_from_clist:Nn \l_tmpa_seq { #3 }
850   \seq_pop_left:NN \l_tmpa_seq \l_tmpa_tl
851   \str_set:Ne \l_tmpa_str {
852     \exp_args:Ne \str_lowercase:n {
853       \tl_use:N \l_tmpa_tl
854     }
855   }
856   \bool_if:NTF \l_lngx_expansion_bool {
857     \str_case:Vn \l__lngx_gls_expansion_case_str {
858       { lowercase } {
859         \text_lowercase:n {
860           \tl_use:c {
861             g_lngx_ \str_use:N \l_tmpa_str _expansion_tl
862           }
863         }
864       }
865       { title~ case~ all } {
866         \text_titlecase_all:n {
867           \tl_use:c {
868             g_lngx_ \str_use:N \l_tmpa_str _expansion_tl
869           }
870         }
871       }
872       { title~ case~ first } {
873         \text_titlecase_first:n {
874           \tl_use:c {
875             g_lngx_ \str_use:N \l_tmpa_str _expansion_tl
876           }
877         }
878       }
879     }
880   } {
881     \use:c { __lngx_gloss_ \str_use:N \l_tmpa_str : }
882   }

```



```

883 \seq_if_empty:NF \l_tmpa_seq {
884   \seq_map_inline:Nn \l_tmpa_seq {
885     \group_begin:
886     \str_clear:N \l_tmpa_str
887     \str_set:Ne \l_tmpa_str {
888       \exp_args:Ne \str_lowercase:n { ##1 }
889     }
890     \bool_if:NTF \l_lngx_expansion_bool {
891       \str_case:Vn \l__lngx_gls_expansion_case_str {
892         { lowercase } {
893           \tl_use:N \l_lngx_expansion_separator_tl
894           \text_lowercase:n {
895             \tl_use:c {
896               g_lngx_ \str_use:N \l_tmpa_str _expansion_tl
897             }
898           }
899         }
900         { title~ case~ all } {
901           \tl_use:N \l_lngx_expansion_separator_tl
902           \text_titlecase_all:n {
903             \tl_use:c {
904               g_lngx_ \str_use:N \l_tmpa_str _expansion_tl
905             }
906           }
907         }
908         { title~ case~ first } {
909           \tl_use:N \l_lngx_expansion_separator_tl
910           \text_titlecase_first:n {
911             \tl_use:c {
912               g_lngx_ \str_use:N \l_tmpa_str _expansion_tl
913             }
914           }
915         }
916       }
917     } {
918       \tl_use:N \l_lngx_gloss_separator_tl
919       \use:c { __lngx_gloss_ \str_use:N \l_tmpa_str : }
920     }
921     \group_end:
922   }
923 }
924 \group_end:
925 }

```

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

`__lngx_dotfill:nnn` For the dotfill between the gloss and the expansion, I create a custom internal command. The code is based on user Jonathan P. Spratte's answer seen here: topanswers.xyz/tex?q=8155#a7758. The dotfill should not be tagged at all and in fact it should be suppressed so that the readers don't go 'dot, dot, dot, dot ...' (Frank has convinced us forever with his TUG 2025 talk).

```

926
927 \cs_new_protected:Npn \__lngx_dotfill:nnn #1#2#3 {
928   %% Courtesy: Jonathan P. Spratte

```

```

929 %% topanswers.xyz/tex?q=8155#a7758 (LPPL)
930 \l__lngx_entry_separator_tl
931 \smallskip
932 \group_begin:
933 \rightskip          = 0pt plus -1fil \prg_do_nothing:
934 \parfillskip        = 0pt plus 1fil \prg_do_nothing:
935 \leftskip           = 1em plus 1fil \prg_do_nothing:
936 \finalhyphendemerits = 0           \prg_do_nothing:
937 \parindent          = -1em         \prg_do_nothing:
938 \bool_if:NT \l__lngx_gls_bold_bool { \textbf } {
939   \lngx_gloss_format:n {
940     #1
941   }
942   \tl_use:N \l__lngx_glossary_separator_tl
943 }
944 #2
945 \bool_if:NT \l__lngx_glosses_page_number_bool {
946   \mode_leave_vertical:
947   \quad
948   \IfDocumentMetadataT {
949     \tag_mc_end:
950     \tag_struct_begin:n {
951       tag                = { Span },
952       actualtext         = { }
953     }
954     \tag_mc_begin:n {
955       tag                = { Span }
956     }
957   }
958   \cleaders
959   \hbox to 0.44em { \hss . \hss }
960   \hskip 0.5cm plus 1fill \prg_do_nothing:
961   \IfDocumentMetadataT {
962     \tag_mc_end:
963     \tag_struct_end:
964     \tag_mc_begin:n { }
965   }
966   \quad
967   \kern 0pt \prg_do_nothing:
968   \em #3
969 }
970 \l__lngx_entry_separator_tl
971 \group_end:
972 }

```

(End of definition for `__lngx_dotfill:nnn`.)

lngx_multicols Here I define the custom multicolumn environment which does nothing if the number of columns is 1.

```

973
974 \NewDocumentEnvironment { lngx_multicols } { m } {
975   \int_compare:nNnTF { 1 } < {
976     \int_use:N \l__lngx_glossary_columns_int
977   } {

```

```

978 \begin { multicol s } {
979 \int _use:N \l __lngx_glossary_columns_int
980 } [ #1 ]
981 } { #1 }
982 \noindent
983 } {
984 \int _compare:nNnT { 1 } < {
985 \int _use:N \l __lngx_glossary_columns_int
986 } {
987 \end { multicol s }
988 }
989 }

```

(End of definition for `lngx_multicol s`. This function is documented on page 21.)

\lngx_gloss_list: Finally we come to the command that prints the glosses. First it sets the boolean for creating the aux file to false.

```

990
991 \cs_new_protected:Npn \lngx_gloss_list: {
992 \bool_gset_false:N \g_lngx_trigger_aux_file_bool

```

I start a group, clear a scratch sequence and set it equal to the sequence that stores the order of the glosses. If the aux file is read, the aux flag is added to the variable, or else it is read on the fly.

```

993 \group_begin:
994 \seq_clear:N \l_tmpa_seq
995 \seq_set_eq:NN \l_tmpa_seq \g__lngx_gls_use_order_seq

```

If the sorting order is set to alphabetical, the sequence needs to get sorted. For that, I use L^AT_EX₃'s mechanism for sorting strings.

```

996 \str_case:Vn \l__lngx_gls_sorting_order_str {
997 { alphabetical } {
998 \seq_sort:Nn \l_tmpa_seq {
999 \str_compare:nNnTF { ##1 } > { ##2 } {
1000 \sort_return_swapped:
1001 } {
1002 \sort_return_same:
1003 }
1004 }
1005 }
1006 }

```

If the style used is `inline`, the glosses come after the each other. That means the default entry separator, i.e., `\par` must be changed. Here I set it to `,~` by default (locally). The separator between the gloss and the entry is defined as a colon followed by a space.

```

1007 \str_if_eq:VnTF \l__lngx_glossary_style_str { inline } {
1008 \group_begin:
1009 \keys_set:nn { lngx_glossing } {
1010 separator = { \c_colon_str \c_space_tl },
1011 entry~ separator = { ,~ }
1012 }

```

Then each item from the sequence is popped (from the left). It is then passed to a string variable to get rid of the catcodes. The string variable is then used in `\MakeLinkTarget*`. The gloss is then printed with its separator in bold shape.

```

1013 \tl_clear:N \l_tmpa_tl
1014 \str_clear:N \l_tmpa_str
1015 \seq_pop_left:NN \l_tmpa_seq \l_tmpa_tl
1016 \str_set:NV \l_tmpa_str \l_tmpa_tl
1017 \IfDocumentMetadataT {
1018   \tag_mc_end:
1019   \tag_struct_begin:n {
1020     tag = { Span },
1021   }
1022   \tag_mc_begin:n {
1023     tag = { Span }
1024   }
1025 }
1026 \MakeLinkTarget * {
1027   lngx_ \str_use:N \l_tmpa_str _glossary
1028 }
1029 \bool_if:NT \l__lngx_gls_bold_bool { \textbf } {
1030   \lngx_gloss_format:n {
1031     \tl_use:N \l_tmpa_tl
1032     \tl_use:N \l_lngx_glossary_separator_tl
1033   }
1034 }
1035 \IfDocumentMetadataT {
1036   \tag_mc_end:
1037   \tag_struct_end:
1038 }

```

Then it is checked in which case the expansion is requested. According to that the `tl` is printed.

```

1039 \str_case:Vn \l__lngx_gls_expansion_case_str {
1040   { lowercase } {
1041     \lngx_expansion_format:n {
1042       \text_lowercase:n {
1043         \tl_use:c {
1044           g_lngx_ \str_use:N \l_tmpa_str _expansion_tl
1045         }
1046       }
1047     }
1048   }
1049   { title~ case~ all } {
1050     \lngx_expansion_format:n {
1051       \text_titlecase_all:n {
1052         \tl_use:c {
1053           g_lngx_ \str_use:N \l_tmpa_str _expansion_tl
1054         }
1055       }
1056     }
1057   }
1058   { title~ case~ first } {
1059     \lngx_expansion_format:n {
1060       \text_titlecase_first:n {
1061         \tl_use:v {
1062           g_lngx_ \str_use:N \l_tmpa_str _expansion_tl
1063         }
1064       }
1065     }
1066   }
1067 }

```

```

1064     }
1065   }
1066 }
1067 }

```

After printing one entry successfully, if there are any more items left in the sequence, they are printed with the same method, but with an entry separator at the beginning.

```

1068   \seq_if_empty:NF \l_tmpa_seq {
1069     \seq_map_inline:Nn \l_tmpa_seq {
1070       \group_begin:
1071       \tl_use:N \l__lngx_entry_separator_tl
1072       \MakeLinkTarget * { lngx_ ##1 _glossary }
1073       \textbf {
1074         \lngx_gloss_format:n {
1075           ##1
1076           \tl_use:N \l_lngx_glossary_separator_tl
1077         }
1078       }
1079       \str_case:Vn \l__lngx_gls_expansion_case_str {
1080         { lowercase } {
1081           \lngx_expansion_format:n {
1082             \text_lowercase:n {
1083               \exp_not:v { g_lngx_ ##1 _expansion_tl }
1084             }
1085           }
1086         }
1087         { title~ case~ all } {
1088           \lngx_expansion_format:n {
1089             \text_titlecase_all:n {
1090               \exp_not:v { g_lngx_ ##1 _expansion_tl }
1091             }
1092           }
1093         }
1094         { title~ case~ first } {
1095           \lngx_expansion_format:n {
1096             \text_titlecase_first:n {
1097               \exp_not:v { g_lngx_ ##1 _expansion_tl }
1098             }
1099           }
1100         }
1101       }
1102       \group_end:
1103     }
1104   }
1105   \group_end:
1106 } {

```

If the style is not `inline`, then the default `block` style is assumed and firstly the word ‘glossary’ is printed in a sectioning command controlled by the keys. The `\glossaryname` command is provided by `babel`. If it is undefined, that means the user hasn’t loaded `babel`. In that case, I define the command with the string `Glossary`.

```

1107   \ProvideDocumentCommand \glossaryname { } { Glossary }

```

Then the `lngx_multicols` environment starts which doesn’t do anything if the number of columns is 1.

```

II08 \begin { lngx_multicols } {
II09 \str_if_eq:VnF \l__lngx_gls_sectioning_str { null } {
II10 \use:e {
II11 \exp_not:N \use:c
II12 { \str_use:N \l__lngx_gls_sectioning_str }
II13 \bool_if:NF \l__lngx_gls_section_number_bool { * }
II14 { \exp_not:N \glossaryname }
II15 }
II16 }
II17 }
II18 \seq_map_inline:Nn \l_tmpa_seq {

```

In this style, even the page numbers are printed along with glosses. We save the page numbers in a temporary sequence which is locally cleared.

```

II19 \group_begin:
II20 \seq_clear:N \l_tmpb_seq
II21 \seq_map_inline:cn { g_lngx_ ##1 _pages_seq } {

```

The pages are hyperlinked with the internal PDF names.

```

II22 \seq_put_right:Ne \l_tmpb_seq { ####1 }
II23 }

```

The page numbers are separated using dotfill. Before the glosses, `\MakeLinkTarget*` is used.

```

II24 \__lngx_dotfill:nnn {
II25 \MakeLinkTarget * { lngx_ ##1 _glossary }
II26 ##1
II27 } {

```

The case of expansion is checked and then the appropriate casing commands are used for expansions.

```

II28 \str_case:Vn \l__lngx_gls_expansion_case_str {
II29 { lowercase } {
II30 \lngx_expansion_format:n {
II31 \text_lowercase:n {
II32 \exp_not:v { g_lngx_ ##1 _expansion_tl }
II33 }
II34 }
II35 }
II36 { title~ case~ all } {
II37 \lngx_expansion_format:n {
II38 \text_titlecase_all:n {
II39 \exp_not:v { g_lngx_ ##1 _expansion_tl }
II40 }
II41 }
II42 }
II43 { title~ case~ first } {
II44 \lngx_expansion_format:n {
II45 \text_titlecase_first:n {
II46 \exp_not:v { g_lngx_ ##1 _expansion_tl }
II47 }
II48 }
II49 }
II50 }
II51 } {

```

The list of page numbers is printed.

```

1152         \seq_use:Nn \l_tmpb_seq { ,~ }
1153     }
1154     \group_end:
1155 }
1156 \end { lngx_multicols }
1157 }
1158 \group_end:
1159 }

```

(End of definition for `\lngx_gloss_list:`. This function is documented on page 20.)

`\listofglosses` Here is the command that defines the user-side command for printing the glosses. It defines the separator by default if not provided. All settings are local in order to avoid leaking. `\l_lngx_separator_tl` is the generic string that is used inside the **separator** key that sets the separator contextually. This command uses the \LaTeX function for printing the glosses.

```

1160
1161 \NewDocumentCommand \listofglosses { 0 { } } {
1162     \group_begin:
1163     \str_set:Nn \l_lngx_separator_str { glossary }
1164     \keys_set:nn { lngx_glossing } {
1165         separator = { \c_colon_str \c_space_tl }
1166     }
1167     \keys_set:nn { lngx_glossing } { #1 }
1168     \lngx_gloss_list:
1169     \group_end:
1170 }
1171 \</glossing>

```

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

```

1172 \*ipa
1173 \ProvidesExplPackage{linguistix-ipa}
1174     {2026-05-15}
1175     {v1.0}
1176     {%
1177     A package for typesetting the IPA
1178     (International Phonetic Alphabet) from
1179     the ‘LinguisTiX’ bundle.%
1180     }

```

Then, I load unicode-math or lua-unicode-math (depending on the engine used), LINGUISTIX-NFSS and LINGUISTIX-BASE (if they are not already loaded).

```

1181
1182 \sys_if_engine luatex:TF {
1183   \IfPackageLoadedF { unicode-math } {
1184     \IfPackageLoadedF { lua-unicode-math } {
1185       \RequirePackage { fontspec, lua-unicode-math }
1186     }
1187   }
1188 } {
1189   \IfPackageLoadedF { unicode-math } {
1190     \RequirePackage { unicode-math }
1191   }
1192 }
1193
1194 \IfPackageLoadedF { linguistix-base } {
1195   \RequirePackage { linguistix-base }
1196 }
1197
1198 \IfPackageLoadedF { linguistix-nfss } {
1199   \RequirePackage { linguistix-nfss }
1200 }
1201
1202 \IfPackageLoadedF { linguistix-fixpex } {
1203   \RequirePackage { linguistix-fixpex }
1204 }

```

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

```

1205
1206 \NewDocumentCommand \ipatext { s m } {
1207   \IfBooleanTF { #1 } {
1208     {
1209       \lngxipa
1210       / #2 /
1211     }
1212   } {
1213     {
1214       \lngxipa
1215       [ #2 ]
1216     }
1217   }
1218 }

```

(End of definition for `\ipatext` and `\ipatext*`. These functions are documented on page 11.)


```

\g__lngx_ipa_main_fonts_prop
\g__lngx_ipa_main_font_features_tl
    ipa upright
ipa upright features
    ipa bold
ipa bold 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.

```

I219
I220 \prop_gclear_new:N \g__lngx_ipa_main_fonts_prop
I221 \tl_gclear_new:N \g__lngx_ipa_main_font_features_tl
I222
I223 \clist_map_inline:nn {
I224   upright,
I225   bold,
I226   italic,
I227   bold~ italic,
I228   slanted,
I229   bold~ slanted,
I230   swash,
I231   bold~ swash,
I232   small~ caps
I233 } {
I234   \group_begin:
I235   \str_clear:N \l_tmpa_str
I236   \str_set:Nn \l_tmpa_str { #1 }
I237   \str_replace_all:Nnn \l_tmpa_str { ~ } { _ }
I238   \tl_gclear_new:c {
I239     g__lngx_ipa_main_features_ \l_tmpa_str _tl
I240   }
I241   \group_end:
I242   \keys_define:nn { lngx_keys } {
I243     ipa~ #1
I244     .code:n = {
I245       \group_begin:
I246       \str_clear:N \l_tmpa_str
I247       \str_set:Nn \l_tmpa_str {
I248         \text_titlecase_all:n { #1 }
I249         Font
I250       }
I251       \str_replace_all:Nnn \l_tmpa_str { ~ } { }
I252       \prop_gput:Nne \g__lngx_ipa_main_fonts_prop
I253         { ipa~ #1 }
I254         { \str_use:N \l_tmpa_str = { ##1 } }
I255       \group_end:
I256     },
I257     ipa~ #1~ features
I258     .code:n = {
I259       \group_begin:
I260       \str_clear:N \l_tmpa_str
I261       \str_clear:N \l_tmpb_str
I262       \str_set:Nn \l_tmpa_str {
I263         \text_titlecase_all:n { #1 }
I264         Features
I265       }
I266       \str_set:Nn \l_tmpb_str { #1 }
I267       \str_replace_all:Nnn \l_tmpa_str { ~ } { }

```

All the keys here are prefixed with the word `ipa` in order to distinguish them from the keys provided by the `LINGUISTICS-FONTS` package. These keys have identical method as their text counterparts, though.

```

1268 \str_replace_all:Nnn \l_tmpb_str { ~ } { _ }
1269 \tl_gput_right:ce {
1270   g__lngx_ipa_main_features_ \l_tmpb_str_tl
1271 } {
1272   \tl_if_empty:cF {
1273     g__lngx_ipa_main_features_ \l_tmpb_str_tl
1274   } { , }
1275   \exp_not:n { ##1 }
1276 }
1277 \prop_gput:Nne \g__lngx_ipa_main_fonts_prop
1278   { ipa~ #1~ features } {
1279   \str_use:N \l_tmpa_str = {
1280     \exp_not:v {
1281       g__lngx_ipa_main_features_ \l_tmpb_str_tl
1282     }
1283   }
1284 }
1285 \group_end:
1286 }
1287 }
1288 }

```

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

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

```

1289
1290 \tl_gclear_new:N \g__lngx_ipa_main_features_extra_tl
1291
1292 \keys_define:nn { lngx_keys } {
1293   ipa~ main~ extra~ features
1294   .code:n = {
1295     \tl_gput_right:Ne \g__lngx_ipa_main_features_extra_tl {
1296       \tl_if_empty:NF \g__lngx_ipa_main_features_extra_tl {
1297         ,
1298       }
1299       \exp_not:n { #1 }
1300     }
1301     \prop_gput:Nne \g__lngx_ipa_main_fonts_prop
1302       { ipa~ main~ extra~ features } {
1303       \exp_not:V \g__lngx_ipa_main_features_extra_tl
1304     }
1305   }
1306 }

```

(End of definition for `ipa main extra features`. This function is documented on page [13.](#))

\g__lngx_ipa_sans_fonts_prop
\g__lngx_ipa_sans_font_features_tl
\g__lngx_ipa_mono_fonts_prop
\g__lngx_ipa_mono_font_features_tl
ipa sans extra features
ipa sans upright
ipa sans upright features
ipa sans bold
ipa sans bold 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 extra features
ipa mono upright
ipa mono upright features
ipa mono bold
ipa mono bold 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

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.

```

I307
I308 \prop_gclear_new:N \g__lngx_ipa_sans_fonts_prop
I309 \tl_gclear_new:N \g__lngx_ipa_sans_font_features_tl
I310
I311 \prop_gclear_new:N \g__lngx_ipa_mono_fonts_prop
I312 \tl_gclear_new:N \g__lngx_ipa_mono_font_features_tl
I313
I314 \clist_map_inline:nn {
I315   sans,
I316   mono
I317 } {
I318   \tl_gclear_new:c {
I319     g__lngx_ipa_ #1 _features_extra_tl
I320   }
I321   \clist_map_inline:nn {
I322     upright,
I323     bold,
I324     italic,
I325     bold~ italic,
I326     slanted,
I327     bold~ slanted,
I328     swash,
I329     bold~ swash,
I330     small~ caps
I331   } {
I332     \group_begin:
I333     \str_clear:N \l_tmpa_str
I334     \str_set:Nn \l_tmpa_str { ##1 }
I335     \str_replace_all:Nnn \l_tmpa_str { ~ } { _ }
I336     \tl_gclear_new:c {
I337       g__lngx_ipa_ #1 _features_ \l_tmpa_str _tl
I338     }
I339     \group_end:
I340     \keys_define:nn { lngx_keys } {
I341       ipa~ #1~ ##1
I342       .code:n          = {
I343         \group_begin:
I344         \str_clear:N \l_tmpa_str
I345         \str_set:Ne \l_tmpa_str {
I346           \text_titlecase_all:n { ##1 }
I347           Font
I348         }
I349         \str_replace_all:Nnn \l_tmpa_str { ~ } { }
I350         \prop_gput:cne { g__lngx_ipa_ #1 _fonts_prop }
I351           { ipa~ #1~ ##1 } {
I352           \str_use:N \l_tmpa_str = {
I353             \exp_not:n { #####1 }
I354           }
I355         }
I356       \group_end:
I357     },
I358     ipa~ #1~ ##1~ features

```

```

I359 .code:n = {
I360 \group_begin:
I361 \str_clear:N \l_tmpa_str
I362 \str_clear:N \l_tmpb_str
I363 \str_set:Ne \l_tmpa_str {
I364 \ipa_titlecase_all:n { ##1 }
I365 Features
I366 }
I367 \str_set:Nn \l_tmpb_str { ##1 }
I368 \str_replace_all:Nnn \l_tmpa_str { ~ } { }
I369 \str_replace_all:Nnn \l_tmpb_str { ~ } { _ }
I370 \tl_gput_right:ce {
I371 g__lngx_ipa_ #1 _features_ \l_tmpb_str _tl
I372 } {
I373 \tl_if_empty:cF {
I374 g__lngx_ipa_ #1 _features_ \l_tmpb_str _tl
I375 } { , }
I376 \exp_not:n { ####1 }
I377 }
I378 \prop_gput:cne { g__lngx_ipa_ #1 _fonts_prop }
I379 { ipa~ #1~ ##1~ features } {
I380 \str_use:N \l_tmpa_str = {
I381 \exp_not:v {
I382 g__lngx_ipa_ #1 _features_ \l_tmpb_str _tl
I383 }
I384 }
I385 }
I386 \group_end:
I387 }
I388 }
I389 }
I390 \keys_define:nn { lngx_keys } {
I391 ipa~ #1~ extra~ features
I392 .code:n = {
I393 \tl_gput_right:ce {
I394 g__lngx_ipa_ #1 _features_extra_tl
I395 } {
I396 \tl_if_empty:cF {
I397 g__lngx_ipa_ #1 _features_extra_tl
I398 } { , }
I399 \exp_not:n { ##1 }
I400 }
I401 \prop_gput:cne { g__lngx_ipa_ #1 _fonts_prop }
I402 { ipa~ #1~ extra~ features } {
I403 \exp_not:v { g__lngx_ipa_ #1 _features_extra_tl }
I404 }
I405 }
I406 }
I407 }

```

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

```

\g__lngx_ipa_main_font_tl
\g__lngx_ipa_sans_font_tl
\g__lngx_ipa_mono_font_tl
ipa main font
ipa sans font
ipa mono font

```

These keys provide keys to set fonts for IPA.

```

I408
I409 \clist_map_inline:nn {
I410     main,
I411     sans,
I412     mono
I413 } {
I414     \keys_define:nn { lngx_keys } {
I415         ipa~ #1~ font
I416         .tl_gset:c          = { g__lngx_ipa_ #1 _font_tl }
I417     }
I418 }

```

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

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.

```

I419
I420 \keys_define:nn { lngx_keys } {
I421     ipa~ newcm
I422     .meta:n          = {
I423         ipa~ main~ font      = { NewCM10-Book.otf },
I424         ipa~ sans~ font     = { NewCMSans10-Book.otf },
I425         ipa~ mono~ font     = { NewCMMono10-Book.otf }
I426     }
I427 }

```

(End of definition for `ipa newcm`. This function is documented on page II.)

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

```

I428
I429 \keys_define:nn { lngx_keys } {
I430     ipa~ newcm~ sans
I431     .meta:n          = {
I432         ipa~ main~ font      = { NewCMSans10-Book.otf },
I433         ipa~ sans~ font     = { NewCMSans10-Book.otf },
I434         ipa~ mono~ font     = { NewCMMono10-Book.otf }
I435     }
I436 }

```

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

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

```

I437
I438 \keys_define:nn { lngx_keys } {
I439     ipa~ newcm~ mono
I440     .meta:n          = {
I441         ipa~ main~ font      = { NewCMMono10-Book.otf },
I442         ipa~ sans~ font     = { NewCMSans10-Book.otf },
I443         ipa~ mono~ font     = { NewCMMono10-Book.otf }
I444     }
I445 }

```

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

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

```

1446
1447 \keys_define:nn { lngx_keys } {
1448   ipa~ newcm~ regular
1449   .meta:n          = {
1450     ipa~ main~ font    = { NewCM10-Regular.otf },
1451     ipa~ sans~ font    = { NewCMSans10-Regular.otf },
1452     ipa~ mono~ font    = { NewCMMono10-Regular.otf }
1453   }
1454 }

```

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

`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.

```

1455
1456 \keys_define:nn { lngx_keys } {
1457   ipa~ newcm~ regular~ sans
1458   .meta:n          = {
1459     ipa~ main~ font    = { NewCMSans10-Regular.otf },
1460     ipa~ sans~ font    = { NewCMSans10-Regular.otf },
1461     ipa~ mono~ font    = { NewCMMono10-Regular.otf }
1462   }
1463 }

```

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

`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.

```

1464
1465 \keys_define:nn { lngx_keys } {
1466   ipa~ newcm~ regular~ mono
1467   .meta:n          = {
1468     ipa~ main~ font    = { NewCMMono10-Regular.otf },
1469     ipa~ sans~ font    = { NewCMSans10-Regular.otf },
1470     ipa~ mono~ font    = { NewCMMono10-Regular.otf }
1471   }
1472 }

```

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

We set the `ipa newcm` key by default.

```

1473
1474 \lngx_set_keys:n {ipa~ newcm}

```

`\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.

`\lngx_main_ipa:`

`lngx_ipa_rm_nfss` These commands set NFSS families that we use later for setting the IPA fonts. These functions and NFSS families are public, but manipulating them has effects (mostly desired) at several other places, so use them with caution.

`\lngx_set_sans_ipa_font:nn`

`\lngx_sans_ipa:`

`lngx_ipa_sf_nfss`

`\lngx_set_mono_ipa_font:nn`

`\lngx_mono_ipa:`

`lngx_ipa_tt_nfss`

```

1475
1476 \cs_new_protected:Npn \lngx_set_main_ipa_font:nn #1#2 {
1477   \setfontfamily \lngx_main_ipa: [

```

```

1478     #1,
1479     NFSSFamily              = { lngx_ipa_rm_nfss }
1480   ] { #2 }
1481 }
1482
1483 \cs_new_protected:Npn \lngx_set_sans_ipa_font:nn #1#2 {
1484   \setfontfamily \lngx_sans_ipa: [
1485     #1,
1486     NFSSFamily              = { lngx_ipa_sf_nfss }
1487   ] { #2 }
1488 }
1489
1490 \cs_new_protected:Npn \lngx_set_mono_ipa_font:nn #1#2 {
1491   \setfontfamily \lngx_mono_ipa: [
1492     #1,
1493     NFSSFamily              = { lngx_ipa_tt_nfss }
1494   ] { #2 }
1495 }
1496
1497 \cs_generate_variant:Nn \lngx_set_main_ipa_font:nn { VV }
1498 \cs_generate_variant:Nn \lngx_set_sans_ipa_font:nn { VV }
1499 \cs_generate_variant:Nn \lngx_set_mono_ipa_font:nn { VV }

```

(End of definition for `\lngx_set_main_ipa_font:nn` and others. These functions are documented on page 21.)

lngx_ipa Here, I create a ‘super font family’ with `\lngx_super_font_family:nn`, a macro provided by `LINGUISTIX-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.

```

1500
1501 \lngx_super_font_family:nn { lngx_ipa } {
1502   rm              = { lngx_ipa_rm_nfss },
1503   sf              = { lngx_ipa_sf_nfss },
1504   tt              = { lngx_ipa_tt_nfss }
1505 }

```

(End of definition for `lngx_ipa`. This function is documented on page 21.)

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

```

1506
1507 \cs_new_protected:Npn \lngx_ipa: {
1508   \lngx softer_super_font_family:n { lngx_ipa }
1509 }
1510
1511 \cs_gset_eq:NN \lngxipa \lngx_ipa:

```

(End of definition for `\lngxipa` and `\lngx_ipa:`. These functions are documented on page 11.)

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

```

1512
1513 \clist_map_inline:nn {
1514   main,

```

```

1515 sans,
1516 mono
1517 } {
1518   \cs_new_protected:cpn {
1519     lngx_build_#1_ipa_font_features:
1520   } {
1521     \prop_map_inline:cn { g__lngx_ipa_#1_fonts_prop } {
1522       \tl_gput_right:cn {
1523         g__lngx_ipa_#1_font_features_tl
1524       } { ####2 }
1525     }
1526   }
1527 }

```

Stylistic set 5 of NewCM is dedicated to linguistics. So we use it here. For correct diacritic placement, we need HarfBuzz renderer. That also is loaded here.

```

1528
1529 \hook_gput_code:nnn { begindocument / before } { . } {
1530   \lngx_set_keys:n {
1531     ipa~ main~ extra~
1532     features          = {
1533       Renderer        = {HarfBuzz},
1534       StylisticSet    = {05}
1535     },
1536     ipa~ sans~ extra~
1537     features          = {
1538       Renderer        = {HarfBuzz},
1539       StylisticSet    = {05}
1540     },
1541     ipa~ mono~ extra~
1542     features          = {
1543       Renderer        = {HarfBuzz},
1544       StylisticSet    = {05}
1545     }
1546   }
1547   \lngx_build_main_ipa_font_features:
1548   \lngx_set_main_ipa_font:VV
1549   \g__lngx_ipa_main_font_features_tl
1550   \g__lngx_ipa_main_font_tl
1551   \lngx_build_sans_ipa_font_features:
1552   \lngx_set_sans_ipa_font:VV
1553   \g__lngx_ipa_sans_font_features_tl
1554   \g__lngx_ipa_sans_font_tl
1555   \lngx_build_mono_ipa_font_features:
1556   \lngx_set_mono_ipa_font:VV
1557   \g__lngx_ipa_mono_font_features_tl
1558   \g__lngx_ipa_mono_font_tl
1559 }
1560 </ipa>

```



```

1561 (*lang)
1562 \ProvidesExplPackage{linguistix-languages}
1563     {2026-05-15}
1564     {v1.0}
1565     {%
1566         An assistant package for automatically
1567         loading fonts and more settings for
1568         languages.%
1569     }

```

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

```

1570
1571 \IfPackageLoadedF { linguistix-base } {
1572     \RequirePackage { linguistix-base }
1573 }

```

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

```

1574
1575 \IfPackageLoadedF { babel } {
1576     \RequirePackage [ provide * = * ] { babel }
1577 }

```

To renew the space between the two lines, `setspace` provides a handy set of commands.

```

1578
1579 \IfPackageLoadedF { setspace } {
1580     \RequirePackage { setspace }
1581 }

```

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

```

1582
1583 \tl_new:N \g_lngx_main_language_tl

```

(End of definition for \g_lngx_main_language_tl. This function is documented on page 21.)

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

```

1584
1585 \clist_new:N \g_lngx_languages_clist

```

(End of definition for \g_lngx_languages_clist. This function is documented on page 21.)

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

`\providelanguage`

```

1586
1587 \cs_new_protected:Npn \lngx_languages:nn #1#2 {
1588     \babelprovide [ #1 ] { #2 }
1589 }
1590
1591 \cs_generate_variant:Nn \lngx_languages:nn { VV }
1592 \cs_gset_eq:NN \providelanguage \lngx_languages:nn

```

(End of definition for \lngx_languages:nn and \providelanguage. These functions are documented on page 21.)

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.

```

1593
1594 \IfPackageLoadedTF { linguistix-fonts } {
1595   \cs_gset_protected:Npn \lngx_set_main_font:nn #1#2 {
1596     \babelfont { rm } [ #1 ] { #2 }
1597   }
1598   \cs_gset_protected:Npn \lngx_set_sans_font:nn #1#2 {
1599     \babelfont { sf } [ #1 ] { #2 }
1600   }
1601   \cs_gset_protected:Npn \lngx_set_mono_font:nn #1#2 {
1602     \babelfont { tt } [ #1 ] { #2 }
1603   }
1604 } {
1605   \cs_new_protected:Npn \lngx_set_main_font:nn #1#2 {
1606     \babelfont { rm } [ #1 ] { #2 }
1607   }
1608   \cs_new_protected:Npn \lngx_set_sans_font:nn #1#2 {
1609     \babelfont { sf } [ #1 ] { #2 }
1610   }
1611   \cs_new_protected:Npn \lngx_set_mono_font:nn #1#2 {
1612     \babelfont { tt } [ #1 ] { #2 }
1613   }
1614 }

```

`\lngx_other_main_font:nnn`
`\lngx_other_sans_font:nnn`
`\lngx_other_mono_font:nnn`

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

```

1615
1616 \cs_gset_protected:Npn \lngx_other_main_font:nnn #1#2#3 {
1617   \babelfont [ #1 ] { rm } [ #2 ] { #3 }
1618 }
1619
1620 \cs_gset_protected:Npn \lngx_other_sans_font:nnn #1#2#3 {
1621   \babelfont [ #1 ] { sf } [ #2 ] { #3 }
1622 }
1623
1624 \cs_gset_protected:Npn \lngx_other_mono_font:nnn #1#2#3 {
1625   \babelfont [ #1 ] { tt } [ #2 ] { #3 }
1626 }
1627
1628 \cs_generate_variant:Nn \lngx_other_main_font:nnn { nee }
1629 \cs_generate_variant:Nn \lngx_other_sans_font:nnn { nee }
1630 \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 20.)

`\lngx_load_languages:n`
`\loadlanguages`

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

```

1631
1632 \cs_new_protected:Npn \lngx_load_languages:n #1 {
1633   \lngx_set_keys:n { languages = { #1 } }

```

```

r634 }
r635
r636 \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 21.)

`\lngx_counter:n` 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.

```

r637
r638 \cs_gset_eq:NN \lngx_counter:n \arabic

```

(End of definition for `\lngx_counter:n`. This function is documented on page 22.)

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

```

r639
r640 \cs_set:Npn \thechapter {
r641   \lngx_counter:n { chapter }
r642 }
r643 \cs_set:Npn \thesection {
r644   \lngx_counter:n { section }
r645 }
r646 \cs_set:Npn \thesubsection {
r647   \lngx_counter:n { subsection }
r648 }
r649 \cs_set:Npn \thesubsubsection {
r650   \lngx_counter:n { subsubsection }
r651 }
r652 \cs_set:Npn \theparagraph {
r653   \lngx_counter:n { section }
r654 }
r655 \cs_set:Npn \thesubparagraph {
r656   \lngx_counter:n { section }
r657 }
r658 \cs_set:Npn \thepage {
r659   \lngx_counter:n { page }
r660 }
r661 \cs_set:Npn \thefigure {
r662   \lngx_counter:n { figure }
r663 }
r664 \cs_set:Npn \thetable {
r665   \lngx_counter:n { table }
r666 }
r667 \cs_set:Npn \thefootnote {
r668   \lngx_counter:n { footnote }
r669 }
r670 \cs_set:Npn \thempfootnote {
r671   \lngx_counter:n { mpfootnote }
r672 }
r673 \cs_set:Npn \theequation {
r674   \lngx_counter:n { equation }
r675 }

```

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

```

r676

```

```

1677 \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).

```

1678
1679 \socket_new_plug:nnn { lngx / native-numbering }
1680 { strict } {
1681   \cs_gset_eq:Nc \lngx_counter:n {
1682     \tl_use:N \g_lngx_main_language_tl counter
1683   }
1684 }

```

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

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 \TeX -logically).

```

1685
1686 \socket_new_plug:nnn { lngx / native-numbering }
1687 { logical } {
1688   \cs_gset_protected:Npn \lngx_counter:n ##1 {
1689     \localecounter { digits } { ##1 }
1690   }
1691 }

```

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

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.

```

1692
1693 \socket_new_plug:nnn { lngx / native-numbering } { off } {
1694   \cs_gset_eq:NN \lngx_counter:n \arabic
1695 }

```

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

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.

```

1696
1697 \keys_define:nn { lngx_keys } {
1698   native~ numbering
1699   .choices:nn      = { strict,logical,off } {
1700     \exp_args:Nee
1701     \socket_assign_plug:nn { lngx / native-numbering } {
1702       \str_use:N \l_keys_choice_str
1703     }
1704     \socket_use:n { lngx / native-numbering }
1705   },

```

Similarly, we set the default value to on.

```

1706 native~ numbering
1707 .default:n          = { strict }
1708 }

```

(End of definition for *native numbering*. This function is documented on page 15.)

\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.

```

1709
1710 \cs_new_protected:Npn \lngx_misc_reset: {
1711   \cs_set:Npn \theenumii { \alph { enumii } }
1712   \cs_set:Npn \labelenumii { ( \theenumii ) }
1713   \cs_set:Npn \theenumiii { \roman { enumiii } }
1714   \cs_set:Npn \labelenumiii { \theenumiii . }
1715   \cs_set:Npn \theenumiv { \Alph { enumiv } }
1716   \cs_set:Npn \labelenumiv { \theenumiv . }
1717   \IfPackageLoadedT { expex } {
1718     \lingset { labeltype = alpha }
1719   }
1720   \cs_gset_eq:NN \emph \textit
1721 }

```

(End of definition for *\lngx_misc_reset:*. This function is documented on page 22.)

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

```

1722
1723 \msg_new:nnn { linguistix-languages } { no_support } {
1724   '#1'~ is~ not~ supported.\
1725   If~ you~ want~ it~ to~ be~ supported,~ please~ report~
1726   to~ the~ maintainers.
1727 }

```

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

```

1728
1729 \keys_define:nn { lngx_keys } {
1730   languages
1731   .code:n          = {

```

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

```

1732   \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.

```

1733   \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.

```

1734   \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.

```

1735 \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.

```

1736 \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.

```

1737 \tl_set:Ne \l_tmpb_tl {
1738   main,

```

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

```

1739   import
1740 }

```

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

```

1741 \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.

```

1742 \file_if_exist:nTF { linguistix - \l_tmpa_tl . sty } {
1743   \exp_args:Ne \RequirePackage
1744     { linguistix - \l_tmpa_tl }
1745 } {

```

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.

```

1746   \msg_warning:nnV { linguistix-languages }
1747     { no_support }
1748     \l_tmpa_tl
1749 }

```

The temporary `tls` are cleared.

```

1750 \tl_clear:N \l_tmpa_tl
1751 \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.

```

1752 \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`.

```

1753 \clist_map_inline:Nn \l_tmpa_clist {
1754   \clist_pop:NN \l_tmpa_clist \l_tmpa_tl
1755   \tl_set:Ne \l_tmpb_tl { import }
1756   \lngx_languages:VV \l_tmpb_tl \l_tmpa_tl
1757   \file_if_exist:nTF {
1758     linguistix - \l_tmpa_tl . sty
1759   } {
1760     \exp_args:Ne \RequirePackage

```

```

1761             { linguistix - \l_tmpa_tl }
1762         } {
1763             \msg_warning:nnV { linguistix-languages }
1764                 { no_ldf }
1765                 \l_tmpa_tl
1766         }
1767         \tl_clear:N \l_tmpa_tl
1768         \tl_clear:N \l_tmpb_tl
1769     }
1770 }
1771 }
1772 }
1773 }
1774 </lang>

```

(End of definition for *languages*. This function is documented on page [I5](#).)

```

1775 <*logos>
1776 \ProvidesExplPackage{linguistix-logos}
1777     {2026-05-15}
1778     {v1.0}
1779     {%
1780     Logos of the ‘LinguisTiX’ bundle.%
1781     }

```

The fontspec package (if not already loaded).

```

1782
1783 \IfPackageLoadedF { fontspec } {
1784     \RequirePackage { fontspec }
1785 }

```

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

```

1786
1787 \newfontfamily \lngx_logo_font: [
1788     UprightFont          = { NewCMUncial10-Book.otf },
1789     UprightFeatures      = {
1790         SizeFeatures      = {
1791             {
1792                 Size          = {-8},
1793                 Font          = {NewCMUncial08-Book.otf}
1794             },
1795             {
1796                 Size          = {8-},
1797                 Font          = {NewCMUncial10-Book.otf}
1798             },
1799         },
1800     },
1801     BoldFont             = { NewCMUncial10-Bold.otf },
1802     BoldFeatures         = {
1803         SizeFeatures      = {
1804             {
1805                 Size          = {-8},
1806                 Font          = {NewCMUncial08-Bold.otf}
1807             },
1808             {
1809                 Size          = {8-},
1810                 Font          = {NewCMUncial10-Bold.otf}
1811             },
1812         },
1813     }
1814 ]{ NewCMUncial10-Book.otf }

```

(End of definition for \lngx_logo_font:. This function is documented on page 22.)

lngx_purple_color The following defines the lngx_purple_color.

```

1815
1816 \color_set:nn { lngx_purple_color } { blue ! 50 ! red }

```

(End of definition for lngx_purple_color. This function is documented on page 22.)

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

```

1817
1818 \NewDocumentCommand \lngxlogo { 0{} } {%
1819   \group_begin:
1820   \lngx_logo_font:
1821   LinguisTi
1822   \color_group_begin:
1823   \color_select:n { lngx_purple_color }
1824   X
1825   \color_group_end:
1826   \IfBlankF { #1 } { - #1 }
1827   \group_end:
1828 }

```

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

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

```

1829
1830 \cs_new:Npn \lngxpkg {
1831   \IfPackageLoadedTF { hyperref } {
1832     \texorpdfstring {
1833       \lngxlogo
1834     } {
1835       LinguisTiX
1836     }
1837   } {
1838     \lngxlogo
1839   }
1840 }

```

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

```

1841
1842 \clist_map_inline:nn {
1843   base,examples,fixpex,fonts,ipa,languages,logos,nfss,
1844   marathi,british,american,english,greek,malayalam,glossing,
1845   leipzig,russian
1846 } {

```

`#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.

```

1847   \cs_new:cpn { lngx #1 logo } {
1848     \texorpdfstring {
1849       \lngxlogo [ #1 ]
1850     } {
1851       LinguisTiX - #1
1852     }
1853   }
1854 }
1855 </logos>

```

LINGUIS*Ti*X-NFSS

Documentation | L^AT_EX₃-interface

```

1856 < *nfss >

```

```

1857 \ProvidesExplPackage{linguistix-nfss}
1858         {2026-05-15}
1859         {v1.0}
1860         {%
1861         An extension to the core NFSS commands
1862         from the ‘LinguisTiX’ bundle.%
1863     }

```

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.

```

1864
1865 \tl_new:N \l__lngx_normalfont_tmp_tl
1866 \tl_new:N \l__lngx_selectfont_tmp_tl
1867 \tl_new:N \l__lngx_family_tmp_tl
1868 \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.

```

1869
1870 \tl_new:N \l_lngx_current_encoding_tl
1871 \tl_new:N \l_lngx_current_meta_family_tl
1872 \tl_new:N \l_lngx_current_super_family_tl
1873 \tl_new:N \l_lngx_current_series_tl
1874 \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
`\c_lngx_default_sfdefault_tl` initialisation can be assumed to have happened. I set some publicly available t_ls here.
`\c_lngx_default_ttdefault_tl`

```

1875
1876 \hook_gput_code:nnn { begindocument / end } { . } {
1877     \tl_const:Ne \c_lngx_default_rmdefault_tl { \rmdefault }
1878     \tl_const:Ne \c_lngx_default_sfdefault_tl { \sfdefault }
1879     \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 22.)

`\l_lngx_current_encoding_tl` First, I set the value `default` for the initial super font family.
`\l_lngx_current_meta_family_tl`
`\l_lngx_current_super_family_tl`
`\l_lngx_current_series_tl`
`\l_lngx_current_shape_tl`

```

1880 \tl_set:Nn \l_lngx_current_super_family_tl { default }
1881
1882 \tl_set:Ne \l_lngx_current_encoding_tl {
1883     \encodingdefault
1884 }

```

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.

```

1884 \tl_set:Ne \l_lngx_current_meta_family_tl {
1885     \@currentmetafamily % new from ltnews42, thanks Frank!
1886 }

```

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

```

1887 \tl_set:Nn \l_lngx_current_series_tl { md }
1888 \tl_set:Nn \l_lngx_current_shape_tl { up }
1889 }

```

(End of definition for `\l_lngx_current_encoding_tl` and others. These functions are documented on page 22.)

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

```

1890
1891 \hook_gput_code:nnn { cmd / selectfont / before } { . } {
1892   \tl_set:Nc \l__lngx_selectfont_tmp_tl { \f@encoding }
1893 }

```

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`.

```

1894
1895 \hook_gput_code:nnn { cmd / selectfont / after } { . } {
1896   \tl_set_eq:NN \l_lngx_current_encoding_tl
1897               \l__lngx_selectfont_tmp_tl
1898   \tl_clear:N   \l__lngx_selectfont_tmp_tl
1899 }

```

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.

```

1900
1901 \hook_gput_code:nnn { cmd / rmfamily / before } { . } {
1902   \tl_set:Nn \l_lngx_current_meta_family_tl { rm }
1903   \tl_set:Nc \l__lngx_family_tmp_tl { \f@encoding }
1904 }
1905
1906 \hook_gput_code:nnn { cmd / rmfamily / after } { . } {
1907   \tl_set:Nn \l_lngx_current_meta_family_tl { rm }
1908   \tl_set_eq:NN \l_lngx_current_encoding_tl
1909               \l__lngx_family_tmp_tl
1910   \tl_clear:N   \l__lngx_family_tmp_tl
1911 }
1912
1913 \hook_gput_code:nnn { cmd / sffamily / before } { . } {
1914   \tl_set:Nn \l_lngx_current_meta_family_tl { sf }
1915   \tl_set:Nc \l__lngx_family_tmp_tl { \f@encoding }
1916 }
1917
1918 \hook_gput_code:nnn { cmd / sffamily / after } { . } {
1919   \tl_set:Nn \l_lngx_current_meta_family_tl { sf }
1920   \tl_set_eq:NN \l_lngx_current_encoding_tl
1921               \l__lngx_family_tmp_tl
1922   \tl_clear:N   \l__lngx_family_tmp_tl
1923 }
1924
1925 \hook_gput_code:nnn { cmd / ttfamily / before } { . } {
1926   \tl_set:Nn \l_lngx_current_meta_family_tl { tt }
1927   \tl_set:Nc \l__lngx_family_tmp_tl { \f@encoding }
1928 }
1929
1930 \hook_gput_code:nnn { cmd / ttfamily / after } { . } {

```

```

1931 \tl_set:Nn \l_lngx_current_meta_family_tl { tt }
1932 \tl_set_eq:NN \l_lngx_current_encoding_tl
1933         \l__lngx_family_tmp_tl
1934 \tl_clear:N \l__lngx_family_tmp_tl
1935 }

```

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

```

1936
1937 \hook_gput_code:nnn { cmd / mdseries / after } { . } {
1938   \tl_set:Nn \l_lngx_current_series_tl { md }
1939 }
1940
1941 \hook_gput_code:nnn { cmd / bfseries / after } { . } {
1942   \tl_set:Nn \l_lngx_current_series_tl { bf }
1943 }

```

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

```

1944
1945 \hook_gput_code:nnn { cmd / upshape / after } { . } {
1946   \tl_set:Nn \l_lngx_current_shape_tl { up }
1947 }
1948
1949 \hook_gput_code:nnn { cmd / itshape / after } { . } {
1950   \tl_set:Nn \l_lngx_current_shape_tl { it }
1951 }
1952
1953 \hook_gput_code:nnn { cmd / scshape / after } { . } {
1954   \tl_set:Nn \l_lngx_current_shape_tl { sc }
1955 }
1956
1957 \hook_gput_code:nnn { cmd / sscshape / after } { . } {
1958   \tl_set:Nn \l_lngx_current_shape_tl { ssc }
1959 }
1960
1961 \hook_gput_code:nnn { cmd / slshape / after } { . } {
1962   \tl_set:Nn \l_lngx_current_shape_tl { sl }
1963 }
1964
1965 \hook_gput_code:nnn { cmd / swshape / after } { . } {
1966   \tl_set:Nn \l_lngx_current_shape_tl { sw }
1967 }
1968
1969 \hook_gput_code:nnn { cmd / ulcshape / after } { . } {
1970   \tl_set:Nn \l_lngx_current_shape_tl { ulc }
1971 }

```

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

```

1972
1973 \prg_new_conditional:Nnn \lngx_if_encoding:n {
1974   p,
1975   T,

```

```

1976 F,
1977 TF
1978 } {
1979 \tl_if_eq:NnTF \l_lngx_current_encoding_tl { #1 } {
1980 \prg_return_true:
1981 } {
1982 \prg_return_false:
1983 }
1984 }
1985

```

(End of definition for `\lngx_if_encoding:nTF`. This function is documented on page 22.)

`\IfEncodingTF` For non- \LaTeX 3 contexts, these simpler alternatives are provided.
`\IfEncodingT`
`\IfEncodingF`

```

1986
1987 \cs_new_eq:NN \IfEncodingTF \lngx_if_encoding:nTF
1988 \cs_new_eq:NN \IfEncodingT \lngx_if_encoding:nT
1989 \cs_new_eq:NN \IfEncodingF \lngx_if_encoding:nF

```

(End of definition for `\IfEncodingTF`, `\IfEncodingT`, and `\IfEncodingF`. These functions are documented on page 18.)

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

```

1990
1991 \prg_new_conditional:Nnn \lngx_if_meta_family:n {
1992 P,
1993 T,
1994 F,
1995 TF
1996 } {
1997 \tl_if_eq:NnTF \l_lngx_current_meta_family_tl { #1 } {
1998 \prg_return_true:
1999 } {
2000 \prg_return_false:
2001 }
2002 }

```

(End of definition for `\lngx_if_meta_family:nTF`. This function is documented on page 22.)

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

```

2003
2004 \cs_new_eq:NN \IfMetaFamilyTF \lngx_if_meta_family:nTF
2005 \cs_new_eq:NN \IfMetaFamilyT \lngx_if_meta_family:nT
2006 \cs_new_eq:NN \IfMetaFamilyF \lngx_if_meta_family:nF

```

(End of definition for `\IfMetaFamilyTF`, `\IfMetaFamilyT`, and `\IfMetaFamilyF`. These functions are documented on page 18.)

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

```

2007
2008 \prg_new_conditional:Nnn \lngx_if_super_family:n {
2009 P,
2010 T,
2011 F,
2012 TF

```

```

2013 } {
2014   \tl_if_eq:NnTF \l_lngx_current_super_family_tl { #1 } {
2015     \prg_return_true:
2016   } {
2017     \prg_return_false:
2018   }
2019 }

```

(End of definition for `\lngx_if_super_family:nTF`. This function is documented on page 22.)

`\IfSuperFamilyTF` User-facing conditionals for super family.

`\IfSuperFamilyT`
`\IfSuperFamilyF`

```

2020
2021 \cs_new_eq:NN \IfSuperFamilyTF \lngx_if_super_family:nTF
2022 \cs_new_eq:NN \IfSuperFamilyT \lngx_if_super_family:nT
2023 \cs_new_eq:NN \IfSuperFamilyF \lngx_if_super_family:nF

```

(End of definition for `\IfSuperFamilyTF`, `\IfSuperFamilyT`, and `\IfSuperFamilyF`. These functions are documented on page 18.)

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

`\lngx_if_series:nTF`

```

2024
2025 \prg_new_conditional:Nnn \lngx_if_series:n {
2026   P,
2027   T,
2028   F,
2029   TF
2030 } {
2031   \tl_if_eq:NnTF \l_lngx_current_series_tl { #1 } {
2032     \prg_return_true:
2033   } {
2034     \prg_return_false:
2035   }
2036 }

```

(End of definition for `\lngx_if_series:nTF`. This function is documented on page 22.)

`\IfSeriesTF` Its user-side macros.

`\IfSeriesT`
`\IfSeriesF`

```

2037
2038 \cs_new_eq:NN \IfSeriesTF \lngx_if_series:nTF
2039 \cs_new_eq:NN \IfSeriesT \lngx_if_series:nT
2040 \cs_new_eq:NN \IfSeriesF \lngx_if_series:nF

```

(End of definition for `\IfSeriesTF`, `\IfSeriesT`, and `\IfSeriesF`. These functions are documented on page 18.)

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

`\lngx_if_shape:nTF`

```

2041
2042 \prg_new_conditional:Nnn \lngx_if_shape:n {
2043   P,
2044   T,
2045   F,
2046   TF
2047 } {
2048   \tl_if_eq:NnTF \l_lngx_current_shape_tl { #1 } {
2049     \prg_return_true:

```

```

2050 } {
2051   \prg_return_false:
2052 }
2053 }

```

(End of definition for `\lngx_if_shape:nTF`. This function is documented on page 22.)

\IfShapeTF User-side macros for the same.

```

\IfShapeT 2054
\IfShapeF 2055 \cs_new_eq:NN \IfShapeTF \lngx_if_shape:nTF
2056 \cs_new_eq:NN \IfShapeT \lngx_if_shape:nT
2057 \cs_new_eq:NN \IfShapeF \lngx_if_shape:nF

```

(End of definition for `\IfShapeTF`, `\IfShapeT`, and `\IfShapeF`. These functions are documented on page 18.)

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.

```

2058
2059 \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: 2060
\lngx_if_meta_family_sf:TF 2061 \clist_map_inline:nn {
\lngx_if_meta_family_tt_p: 2062   rm,
2063   sf,
2064   tt
2065 } {
2066   \prg_new_conditional:cnn { lngx_if_meta_family_ #1 : } {
2067     p, T, F, TF
2068   } {
2069     \tl_if_eq:NnTF \l_lngx_current_meta_family_tl { #1 } {
2070       \prg_return_true:
2071     } {
2072       \prg_return_false:
2073     }
2074   }
2075 }

```

(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 23.)

\lngx_if_series_md_p: Separate conditionals for both the series.

```

\lngx_if_series_md:TF 2076
\lngx_if_series_bf_p: 2077 \clist_map_inline:nn {
\lngx_if_series_bf:TF 2078   md,
2079   bf
2080 } {
2081   \prg_new_conditional:cnn { lngx_if_series_ #1 : } {
2082     p, T, F, TF
2083   } {
2084     \tl_if_eq:NnTF \l_lngx_current_series_tl { #1 } {
2085       \prg_return_true:
2086     } {

```

```

2087     \prg_return_false:
2088   }
2089 }
2090 }

```

(End of definition for `\lngx_if_series_md:TF` and `\lngx_if_series_bf:TF`. These functions are documented on page 23.)

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

```

\lngx_if_shape_up_p:TF
\lngx_if_shape_it_p:TF
\lngx_if_shape_it:TF
\lngx_if_shape_sc_p:TF
\lngx_if_shape_sc:TF
\lngx_if_shape_ssc_p:TF
\lngx_if_shape_ssc:TF
\lngx_if_shape_sl_p:TF
\lngx_if_shape_sl:TF
\lngx_if_shape_sw_p:TF
\lngx_if_shape_sw:TF
\lngx_if_shape_ulc_p:TF
\lngx_if_shape_ulc:TF
2091 \clist_map_inline:nn {
2092   up,
2093   it,
2094   sc,
2095   ssc,
2096   sl,
2097   sw,
2098   ulc
2099 } {
2100   \prg_new_conditional:cnn { lngx_if_shape_ #1 : } {
2101     p, T, F, TF
2102   } {
2103     \tl_if_eq:NnTF \l_lngx_current_shape_tl { #1 } {
2104       \prg_return_true:
2105     } {
2106       \prg_return_false:
2107     }
2108   }
2109 }
2110 }

```

(End of definition for `\lngx_if_shape_up:TF` and others. These functions are documented on page 23.)

These keys are used in the argument of `\lngx_super_font_family:nn`. This is why they are separated from the set `lngx_keys`. We create new tls using these keys that save the `rm`, `sf` and `tt` defaults of the new super font family. `\l__lngx_nfss_tmp_tl` is defined by the command that creates the super font family.

```

2111 \clist_map_inline:nn {
2112   rm,
2113   sf,
2114   tt
2115 } {
2116   \keys_define:nn { lngx_nfss } {
2117     #1
2118     .code:n = {
2119       \tl_gclear_new:c {
2120         g_lngx_ \l__lngx_nfss_tmp_tl _ #1 default _tl
2121       }
2122       \tl_gset:cn {
2123         g_lngx_ \l__lngx_nfss_tmp_tl _ #1 default _tl
2124       } { ##1 }
2125     }
2126   }
2127 }
2128 }

```


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

```
2129
2130 \cs_new_protected:Npn \lngx_super_font_family:nn #1#2 {
2131   \tl_set:Nx \l__lngx_nfss_tmp_tl { #1 }
```

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

```
2132   \keys_set:nn { lngx_nfss } { #2 }
2133   \tl_clear:N \l__lngx_nfss_tmp_tl
2134 }
2135
2136 \cs_gset_eq:NN \superfontfamily
2137               \lngx_super_font_family:nn
```

(End of definition for `\lngx_super_font_family:nn` and `\superfontfamily`. These functions are documented on page 23.)

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

```
2138
2139 \cs_new_protected:Npn \lngx_soft_super_font_family:nn #1#2 {
2140   \tl_set:Nx \l__lngx_current_super_family_tl { #1 }
```

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

```
2141   \clist_map_inline:nn {
2142     rm,
2143     sf,
2144     tt
2145   } {
2146     \tl_if_empty:cF { g_lngx_ #1 _ ##1 default_tl } {
2147       \cs_set:cpe { ##1 default } {
2148         \tl_use:c { g_lngx_ #1 _ ##1 default _tl }
2149       }
2150     }
2151   }
```

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

```
2152   \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.

```
2153   \clist_map_inline:nn { #2 } {
2154     \str_case:nn { ##1 } {
2155       { encoding } {
2156         \exp_args:NV \fontencoding
2157           \l__lngx_current_encoding_tl
2158       }
2159       { family } {
2160         \use:c {
2161           \l__lngx_current_meta_family_tl family
```

```

2162     }
2163     \exp_args:NV \fontencoding
2164                 \l_lngx_current_encoding_tl
2165     \selectfont
2166   }
2167   { series } {
2168     \use:c {
2169       \l_lngx_current_series_tl series
2170     }
2171   }
2172   { shape } {
2173     \use:c {
2174       \l_lngx_current_shape_tl shape
2175     }
2176   }
2177 }
2178 }
2179 }
2180
2181 \cs_gset_eq:NN \softsuperfontfamily
2182               \lngx_soft_super_font_family:nn

```

(End of definition for `\lngx_soft_super_font_family:nn` and `\softsuperfontfamily`. These functions are documented on page 23.)

`\lngx softer_super_font_family:n` This function excludes the encoding and resets all the other attributes. It comes with a user-side macro.

`\softersuperfontfamily`

```

2183
2184 \cs_new_protected:Npn \lngx softer_super_font_family:n #1 {
2185   \lngx_soft_super_font_family:nn { #1 } {
2186     family,
2187     series,
2188     shape
2189   }
2190 }
2191
2192 \cs_gset_eq:NN \softersuperfontfamily
2193               \lngx softer_super_font_family:n

```

(End of definition for `\lngx softer_super_font_family:n` and `\softersuperfontfamily`. These functions are documented on page 23.)

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

`\softestsuperfontfamily`

```

2194
2195 \cs_new_protected:Npn \lngx_softest_super_font_family:n #1 {
2196   \lngx_soft_super_font_family:nn { #1 } {
2197     encoding,
2198     family,
2199     series,
2200     shape
2201   }
2202 }
2203
2204 \cs_gset_eq:NN \softestsuperfontfamily
2205               \lngx_softest_super_font_family:n

```

(End of definition for `\lngx_softest_super_font_family:n` and `\softestsuperfontfamily`. These functions are documented on page 23.)

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

```

2206
2207 \cs_new_protected:Npn \lngx_soft_normal_font:n #1 {
2208   \tl_set:Nx \l_lngx_current_super_family_tl { default }
2209   \clist_map_inline:nn {
2210     rm,
2211     sf,
2212     tt
2213   } {
2214     \cs_set:cpe { ##1 default } {
2215       \tl_use:c { c_lngx_default_ ##1 default _tl }
2216     }
2217   }
2218   \normalfont
2219   \clist_map_inline:nn { #1 } {
2220     \str_case:nn { ##1 } {
2221       { encoding } {
2222         \exp_args:NV \fontencoding
2223           \l_lngx_current_encoding_tl
2224       }
2225       { family } {
2226         \use:c {
2227           \l_lngx_current_meta_family_tl family
2228         }
2229         \exp_args:NV \fontencoding
2230           \l_lngx_current_encoding_tl
2231         \selectfont
2232       }
2233       { series } {
2234         \use:c {
2235           \l_lngx_current_series_tl series
2236         }
2237       }
2238       { shape } {
2239         \use:c {
2240           \l_lngx_current_shape_tl shape
2241         }
2242       }
2243     }
2244   }
2245 }
2246
2247 \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 23.)

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

```

2248
2249 \cs_new_protected:Npn \lngx_softer_normal_font: {

```

```

2250 \lmgx_soft_normal_font:n {
2251     family,
2252     series,
2253     shape
2254 }
2255 }
2256
2257 \cs_gset_eq:NN \softernormalfont \lmgx_softer_normal_font:

```

(End of definition for `\lmgx_softer_normal_font:` and `\softernormalfont`. These functions are documented on page 23.)

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

```

2258
2259 \cs_new_protected:Npn \lmgx_softest_normal_font: {
2260     \lmgx_soft_normal_font:n {
2261         encoding,
2262         family,
2263         series,
2264         shape
2265     }
2266 }
2267
2268 \cs_gset_eq:NN \softestnormalfont \lmgx_softest_normal_font:

```

(End of definition for `\lmgx_softest_normal_font:` and `\softestnormalfont`. These functions are documented on page 23.)

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

```

2269 \cs_new:Npn \CurrentEncoding {
2270     \tl_use:N \l_lmgx_current_encoding_tl
2271 }
2272 \cs_new:Npn \CurrentMetaFamily {
2273     \tl_use:N \l_lmgx_current_meta_family_tl
2274 }
2275 \cs_new:Npn \CurrentSuperFamily {
2276     \tl_use:N \l_lmgx_current_super_family_tl
2277 }
2278 \cs_new:Npn \CurrentSeries {
2279     \tl_use:N \l_lmgx_current_series_tl
2280 }
2281 \cs_new:Npn \CurrentShape {
2282     \tl_use:N \l_lmgx_current_shape_tl
2283 }
2284 \</nfss>

```

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

References

- Bringhurst, 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).

Index

The *italic* numbers denote the pages where the corresponding entry is described, numbers underlined point to the definition, all others indicate the places where it is used.

Symbols

`\` 60, 61, 62, 63, 1724

A

`\addto` 22
`\Alph` 1715
`\alph` 1711
`\arabic` 67, 68, 1638, 1694
`\AssignTaggingSocketPlug` 656

B

`\babelfont` 1596, 1599, 1602, 1606, 1609, 1612, 1617, 1621, 1625
`\babelprovide` 1588
`\begin` 978, 1108
 bool commands:
 `\bool_gset_false:N` 992
 `\bool_gset_true:N` 142
 `\bool_if:NTF` 508, 515, 517, 524, 526, 856, 890, 938, 945, 1029, 1113
 `\bool_new:N` 136, 137, 567
 `\bool_set_true:N` 840
`bourbaki'semptyset` 7, 135

C

`\cleaders` 958
 clist commands:
 `\clist_gset:Nn` 1732
 `\clist_if_empty:NTF` 1734, 1752
 `\clist_map_inline:Nn` 1753
 `\clist_map_inline:nn` 157, 248, 255, 343, 474, 1223, 1314, 1321, 1409, 1513, 1842, 2061, 2077, 2092, 2112, 2141, 2153, 2209, 2219
 `\clist_new:N` 1585
 `\clist_pop:NN` 1735, 1754
 `\clist_set_eq:NN` 1733
 `\l_tmpa_clist` 1733, 1734, 1735, 1752, 1753, 1754
 color commands:
 `\color_group_begin:` 1822
 `\color_group_end:` 1825
 `\color_select:n` 1823
 `\color_set:nn` 1816
 columns 10, 688
 cs commands:
 `\cs_generate_variant:Nn` . 468, 469, 470, 471, 472, 1497, 1498, 1499, 1591, 1628, 1629, 1630, 2059
 `\cs_gset_eq:NN` . . 48, 82, 83, 830, 1511, 1592, 1636, 1638, 1681, 1694, 1720, 2136, 2181, 2192, 2204, 2247, 2257, 2268

\cs_gset_protected:Npn	1595, 1598, 1601, 1616, 1620, 1624, 1688
\cs_if_exist:NTF	795
\cs_new:Npn	1830, 1847, 2269, 2272, 2275, 2278, 2281
\cs_new_eq:NN	1987, 1988, 1989, 2004, 2005, 2006, 2021, 2022, 2023, 2038, 2039, 2040, 2055, 2056, 2057
\cs_new_protected:Npn	47, 44, 443, 446, 449, 454, 458, 479, 490, 498, 746, 759, 927, 991, 1476, 1483, 1490, 1507, 1518, 1587, 1605, 1608, 1611, 1632, 1710, 2130, 2139, 2184, 2195, 2207, 2249, 2259
\cs_set:Npe	2147, 2214
\cs_set:Npn	600, 1640, 1643, 1646, 1649, 1652, 1655, 1658, 1661, 1664, 1667, 1670, 1673, 1711, 1712, 1713, 1714, 1715, 1716
\cs_set_eq:NN	605
\cs_undefine:N	833
\CurrentEncoding	18, 2269
\CurrentMetaFamily	18, 2269
\CurrentSeries	18, 2269
\CurrentShape	18, 2269
\CurrentSuperFamily	18, 2275

D

\DeclareMathVersion	460
dim commands:	
\dim_zero_new:N	571, 572, 573, 574
\DocumentMetadata	46

E

\em	968
\emph	1720
\encodingdefault	1882
\end	987, 1156
entry_separator	10, 719
exp commands:	
\exp_args:Ne	69, 618, 632, 770, 793, 852, 888, 1743, 1760
\exp_args:Nee	46, 763, 1700
\exp_args:NV	2156, 2163, 2222, 2229
\exp_not:N	801, 812, 1111, 1114
\exp_not:n	209, 214, 233, 237, 287, 310, 315, 333, 337, 634, 772, 1083, 1090, 1097, 1132, 1139, 1146, 1275, 1280, 1299, 1303, 1353, 1376, 1381, 1399, 1403
expansion	9, 736
expansion_case	9, 672

F

file commands:	
\file_if_exist:nTF	1742, 1757
\finalhyphendemerits	936
\fontencoding	2156, 2163, 2222, 2229
format	9, 657

G

\GetDocumentProperties	727
------------------------	-----

<code>\gla</code>	82, 83
<code>gloss</code>	9, <u>730</u>
<code>\glossaryname</code>	53, <u>1107</u> , <u>1114</u>
<code>\glx</code>	8, <u>10</u> , <u>836</u>
<code>\glx*</code>	8, <u>10</u>
group commands:	
<code>\group_begin:</code>	168, 179, 193, 266, 277, 294, 629, 747, 769, 798, 838, 885, 932, 993, <u>1008</u> , <u>1070</u> , <u>1119</u> , <u>1162</u> , <u>1234</u> , <u>1245</u> , <u>1259</u> , <u>1332</u> , <u>1343</u> , <u>1360</u> , <u>1819</u>
<code>\group_end:</code>	175, 189, 219, 273, 290, 320, 647, 787, 818, 827, 921, 924, 971, <u>1102</u> , <u>1105</u> , <u>1154</u> , <u>1158</u> , <u>1169</u> , <u>1241</u> , <u>1255</u> , <u>1285</u> , <u>1339</u> , <u>1356</u> , <u>1386</u> , <u>1827</u>

H

<code>\hbox</code>	<u>131</u> , <u>959</u>
hook commands:	
<code>\hook_gput_code:nnn</code>	67, 81, 506, <u>1529</u> , <u>1876</u> , <u>1891</u> , <u>1895</u> , <u>1901</u> , <u>1906</u> , <u>1913</u> , <u>1918</u> , <u>1925</u> , <u>1930</u> , <u>1937</u> , <u>1941</u> , <u>1945</u> , <u>1949</u> , <u>1953</u> , <u>1957</u> , <u>1961</u> , <u>1965</u> , <u>1969</u>
<code>\hskip</code>	960
<code>\hss</code>	959
<code>\hyperlink</code>	645, 785
<code>\hypersetup</code>	632, 770

I

<code>\IfBlankF</code>	<u>1826</u>
<code>\IfBooleanT</code>	839
<code>\IfBooleanTF</code>	<u>1207</u>
<code>\IfDocumentMetadataT</code>	948, 961, <u>1017</u> , <u>1035</u>
<code>\IfDocumentMetadataTF</code>	762
<code>\IfEncodingF</code>	<u>18</u> , <u>1986</u>
<code>\IfEncodingT</code>	<u>18</u> , <u>1986</u>
<code>\IfEncodingTF</code>	<u>18</u> , <u>1986</u>
<code>\IfMetaFamilyF</code>	<u>18</u> , <u>2003</u>
<code>\IfMetaFamilyT</code>	<u>18</u> , <u>2003</u>
<code>\IfMetaFamilyTF</code>	<u>18</u> , <u>2003</u>
<code>\IfPackageLoadedF</code>	10, 13, <u>16</u> , <u>19</u> , 22, 25, 28, 31, 72, <u>106</u> , <u>111</u> , <u>112</u> , <u>117</u> , <u>122</u> , <u>442</u> , <u>507</u> , <u>563</u> , <u>1183</u> , <u>1184</u> , <u>1189</u> , <u>1194</u> , <u>1198</u> , <u>1202</u> , <u>1571</u> , <u>1575</u> , <u>1579</u> , <u>1783</u>
<code>\IfPackageLoadedT</code>	459, 547, <u>1717</u>
<code>\IfPackageLoadedTF</code>	68, 70, 628, 768, <u>1594</u> , <u>1831</u>
<code>\IfPDFManagementActiveT</code>	585, 637, 725, 775
<code>\IfSeriesF</code>	<u>18</u> , <u>2037</u>
<code>\IfSeriesT</code>	<u>18</u> , <u>2037</u>
<code>\IfSeriesTF</code>	<u>18</u> , <u>2037</u>
<code>\IfShapeF</code>	<u>18</u> , <u>2054</u>
<code>\IfShapeT</code>	<u>18</u> , <u>2054</u>
<code>\IfShapeTF</code>	<u>18</u> , <u>2054</u>
<code>\IfSuperFamilyF</code>	<u>18</u> , <u>2020</u>
<code>\IfSuperFamilyT</code>	<u>18</u> , <u>2020</u>
<code>\IfSuperFamilyTF</code>	<u>18</u> , <u>2020</u>

int commands:	
\int_compare:nNnTF	975, 984
\int_gincr:N	792
\int_gzero_new:N	575
\int_use:N	794, 796, 803, 814, 976, 979, 985
ipa_bold	<i>II</i> , <u>1219</u>
ipa_bold_features	<i>II</i> , <u>1219</u>
ipa_bold_italic	<i>I2</i> , <u>1219</u>
ipa_bold_italic_features	<i>I2</i> , <u>1219</u>
ipa_bold_slanted	<i>I2</i> , <u>1219</u>
ipa_bold_slanted_features	<i>I2</i> , <u>1219</u>
ipa_bold_swash	<i>I2</i> , <u>1219</u>
ipa_bold_swash_features	<i>I2</i> , <u>1219</u>
ipa commands:	
\ipa_titlecase_all:n	<i>I364</i>
ipa_italic	<i>I2</i> , <u>1219</u>
ipa_italic_features	<i>I2</i> , <u>1219</u>
ipa_main_extra_features	<i>I3</i> , <u>1289</u>
ipa_main_font	<i>II</i> , <u>1408</u>
ipa_mono_bold	<i>I2</i> , <u>1307</u>
ipa_mono_bold_features	<i>I2</i> , <u>1307</u>
ipa_mono_bold_italic	<i>I2</i> , <u>1307</u>
ipa_mono_bold_italic_features	<i>I2</i> , <u>1307</u>
ipa_mono_bold_slanted	<i>I2</i> , <u>1307</u>
ipa_mono_bold_slanted_features	<i>I3</i> , <u>1307</u>
ipa_mono_bold_swash	<i>I3</i> , <u>1307</u>
ipa_mono_bold_swash_features	<i>I3</i> , <u>1307</u>
ipa_mono_extra_features	<i>I3</i> , <u>1307</u>
ipa_mono_font	<i>I2</i> , <u>1408</u>
ipa_mono_italic	<i>I2</i> , <u>1307</u>
ipa_mono_italic_features	<i>I2</i> , <u>1307</u>
ipa_mono_slanted	<i>I2</i> , <u>1307</u>
ipa_mono_slanted_features	<i>I2</i> , <u>1307</u>
ipa_mono_small_caps	<i>I3</i> , <u>1307</u>
ipa_mono_small_caps_features	<i>I3</i> , <u>1307</u>
ipa_mono_swash	<i>I3</i> , <u>1307</u>
ipa_mono_swash_features	<i>I3</i> , <u>1307</u>
ipa_mono_upright	<i>I2</i> , <u>1307</u>
ipa_mono_upright_features	<i>I2</i> , <u>1307</u>
ipa_newcm	<i>II</i> , <u>1419</u>
ipa_newcm_mono	<i>II</i> , <u>1437</u>
ipa_newcm_regular	<i>II</i> , <u>1446</u>
ipa_newcm_regular_mono	<i>II</i> , <u>1464</u>
ipa_newcm_regular_sans	<i>II</i> , <u>1455</u>

<code>ipa_newcm_sans</code>	II, 1428
<code>ipa_sans_bold</code>	I2, 1307
<code>ipa_sans_bold_features</code>	I2, 1307
<code>ipa_sans_bold_italic</code>	I2, 1307
<code>ipa_sans_bold_italic_features</code>	I2, 1307
<code>ipa_sans_bold_slanted</code>	I2, 1307
<code>ipa_sans_bold_slanted_features</code>	I2, 1307
<code>ipa_sans_bold_swash</code>	I2, 1307
<code>ipa_sans_bold_swash_features</code>	I2, 1307
<code>ipa_sans_extra_features</code>	I3, 1307
<code>ipa_sans_font</code>	I2, 1408
<code>ipa_sans_italic</code>	I2, 1307
<code>ipa_sans_italic_features</code>	I2, 1307
<code>ipa_sans_slanted</code>	I2, 1307
<code>ipa_sans_slanted_features</code>	I2, 1307
<code>ipa_sans_small_caps</code>	I2, 1307
<code>ipa_sans_small_caps_features</code>	I2, 1307
<code>ipa_sans_swash</code>	I2, 1307
<code>ipa_sans_swash_features</code>	I2, 1307
<code>ipa_sans_upright</code>	I2, 1307
<code>ipa_sans_upright_features</code>	I2, 1307
<code>ipa_slanted</code>	I2, 1219
<code>ipa_slanted_features</code>	I2, 1219
<code>ipa_small_caps</code>	I2, 1219
<code>ipa_small_caps_features</code>	I2, 1219
<code>ipa_swash</code>	I2, 1219
<code>ipa_swash_features</code>	I2, 1219
<code>ipa_upright</code>	II, 1219
<code>ipa_upright_features</code>	II, 1219
<code>\ipatext</code>	II, 1205
<code>\ipatext*</code>	II, 1205

K

<code>\kern</code>	I29, I30, I31, I32, 967
keys commands:	
<code>\l_keys_choice_str</code>	I43, 668, 677, 684, I702
<code>\keys_define:nn</code>	I39, I76, 226, 274, 324, 348, 360, 371, 382, 393, 404, 415, 426, 658, 731, I242, I292, I340, I390, I414, I420, I429, I438, I447, I456, I465, I697, I729, 2II7
<code>\keys_set:nn</code>	45, 743, 842, 846, I009, II64, II67, 2I32

L

<code>\label</code>	46, 794
<code>\labelenumii</code>	I712
<code>\labelenumiii</code>	I714
<code>\labelenumiv</code>	I716

languages	15, 1728
\LaTeX	5, 125
\leftskip	935
\lingset	1718
\linguistix	5, 19, 47
link_color	9, 661
\listofglosses	8, 10, 20, 1160
lngx commands:	
\g_lngx_bourbaki_bool	20, 135
\lngx_build_main_ipa_font_features:	1547
\lngx_build_mono_ipa_font_features:	1555
\lngx_build_sans_ipa_font_features:	1551
\lngx_counter:n	15, 22, 67, 68, 1637, 1638, 1641, 1644, 1647, 1650, 1653, 1656, 1659, 1662, 1665, 1668, 1671, 1674, 1681, 1688, 1694
\l_lngx_current_encoding_tl	22, 1870, 1880, 1896, 1908, 1920, 1932, 1979, 2157, 2164, 2223, 2230, 2270
\l_lngx_current_meta_family_tl	22, 1871, 1880, 1902, 1907, 1914, 1919, 1926, 1931, 1997, 2069, 2161, 2227, 2273
\l_lngx_current_series_tl	22, 1873, 1880, 1938, 1942, 2031, 2084, 2169, 2235, 2279
\l_lngx_current_shape_tl	22, 1874, 1880, 1946, 1950, 1954, 1958, 1962, 1966, 1970, 2048, 2104, 2174, 2240, 2282
\l_lngx_current_super_family_tl	22, 1872, 1880, 2014, 2140, 2208, 2276
\c_lngx_default_rmdefault_tl	22, 1875
\c_lngx_default_sfdefault_tl	22, 1875
\c_lngx_default_ttdefault_tl	22, 1875
\l_lngx_expansion_bool	567, 840, 856, 890
\lngx_expansion_format:n	20, 736, 737, 1041, 1050, 1059, 1081, 1088, 1095, 1130, 1137, 1144
\l_lngx_expansion_separator_tl	569, 893, 901, 909
\lngx_gloss_format:n	20, 644, 649, 730, 733, 784, 789, 939, 1030, 1074
\g_lngx_gloss_link_color_str	41
\lngx_gloss_list:	20, 990, 991, 1168
\lngx_gloss_new:nn	20, 47, 745, 746, 830, 834
\l_lngx_gloss_separator_tl	568, 918
\l_lngx_glossary_separator_tl	570, 942, 1032, 1076
\l_lngx_gls_language_str	576, 726
\l_lngx_i_hack_dim	574
\l_lngx_i_have_dim	571
\l_lngx_i_need_dim	572
\lngx_if_encoding:n	1973
\lngx_if_encoding:nTF	22, 1972, 1987, 1988, 1989
\lngx_if_encoding:p:n	22, 1972
\lngx_if_meta_family:n	1991
\lngx_if_meta_family:nTF	22, 1990, 2004, 2005, 2006
\lngx_if_meta_family:p:n	22, 1990
\lngx_if_meta_family_rm:TF	23, 2060

\lngx_if_meta_family_rm_p:	23, 2060
\lngx_if_meta_family_sf:TF	23, 2060
\lngx_if_meta_family_sf_p:	23, 2060
\lngx_if_meta_family_tt:TF	23, 2060
\lngx_if_meta_family_tt_p:	23, 2060
\lngx_if_series:n	2025
\lngx_if_series:nTF	22, 2024, 2038, 2039, 2040
\lngx_if_series_bf:TF	23, 2076
\lngx_if_series_bf_p:	23, 2076
\lngx_if_series_md:TF	23, 2076
\lngx_if_series_md_p:	23, 2076
\lngx_if_series_p:n	22, 2024
\lngx_if_shape:n	2042
\lngx_if_shape:nTF	22, 2041, 2055, 2056, 2057
\lngx_if_shape_it:TF	23, 2091
\lngx_if_shape_it_p:	23, 2091
\lngx_if_shape_p:n	22, 2041
\lngx_if_shape_sc:TF	23, 2091
\lngx_if_shape_sc_p:	23, 2091
\lngx_if_shape_sl:TF	23, 2091
\lngx_if_shape_sl_p:	23, 2091
\lngx_if_shape_ssc:TF	23, 2091
\lngx_if_shape_ssc_p:	23, 2091
\lngx_if_shape_sw:TF	23, 2091
\lngx_if_shape_sw_p:	23, 2091
\lngx_if_shape_ulc:TF	23, 2091
\lngx_if_shape_ulc_p:	23, 2091
\lngx_if_shape_up:TF	23, 2091
\lngx_if_shape_up_p:	23, 2091
\lngx_if_super_family:n	2008
\lngx_if_super_family:nTF	22, 2007, 2021, 2022, 2023
\lngx_if_super_family_p:n	22, 2007
lngx_ipa	21, 1500
\lngx_ipa:	21, 1506, 1507, 1511
lngx_ipa_rm_nfss	21, 1475
lngx_ipa_sf_nfss	21, 1475
lngx_ipa_tt_nfss	21, 1475
\lngx_languages:nn	21, 1586, 1587, 1591, 1592, 1741, 1756
\g_lngx_languages_clist	21, 1584, 1732, 1733
\lngx_load_languages:n	21, 1631, 1632, 1636
\lngx_logo_font:	22, 1786, 1787, 1820
\lngx_main_ipa:	21, 1475, 1477
\g_lngx_main_language_tl	21, 1582, 1682, 1736
\lngx_misc_reset:	22, 1709, 1710

\ngx_mono_ipa: 2I, 1475, 1491
ngx_multicols 2I, 973
\g_ngx_old_style_bool 20, 135, 515, 524
\g_ngx_old_style_one_bool 20, 135, 517, 526
\ngx_other_main_font:nnn 20, 1615, 1616, 1628
\ngx_other_mono_font:nnn 20, 1615, 1624, 1630
\ngx_other_sans_font:nnn 20, 1615, 1620, 1629
ngx_purple_color 22, 1815
\l_ngx_remain_dim 573
\ngx_sans_ipa: 2I, 1475, 1484
\l_ngx_separator_tl 55
\ngx_set_keys:n 19, 43, 44, 48, 437, 509, 512, 1474, 1530, 1633
\ngx_set_main_font:nn 20, 441, 443, 468, 533, 1595, 1605
\ngx_set_main_ipa_font:nn 2I, 1475, 1476, 1497, 1548
\ngx_set_math_bold_font:nn 458, 472, 549
\ngx_set_math_font:nn 20, 441, 454, 471, 545
\ngx_set_mono_font:nn 20, 441, 449, 470, 541, 1601, 1611
\ngx_set_mono_ipa_font:nn 2I, 1475, 1490, 1499, 1556
\ngx_set_sans_font:nn 20, 441, 446, 469, 537, 1598, 1608
\ngx_set_sans_ipa_font:nn 2I, 1475, 1483, 1498, 1552
\ngx_soft_normal_font:n 23, 2206, 2207, 2247, 2250, 2260
\ngx_soft_super_font_family:nn 23, 2138, 2139, 2182, 2185, 2196
\ngx_softer_normal_font: 23, 2248, 2249, 2257
\ngx_softer_super_font_family:n 23, 1508, 2183, 2184, 2193
\ngx_softest_normal_font: 23, 2258, 2259, 2268
\ngx_softest_super_font_family:n 23, 2194, 2195, 2205
\ngx_super_font_family:nn 23, 1501, 2129, 2130, 2137
\g_ngx_trigger_aux_file_bool 992
ngx internal commands:
\g_ngx_bold_math_font_features_tl 473
__ngx_build_bold_math_font_features: 473
__ngx_build_main_font_features: 473, 532
__ngx_build_math_bold_features: 498, 548
__ngx_build_math_features: 490, 544
__ngx_build_math_font_features: 473
__ngx_build_mono_font_features: 473, 540
__ngx_build_sans_font_features: 473, 536
__ngx_dotfill:nnn 926, 927, 1124
\l_ngx_entry_separator_tl 719, 930, 970, 1071
\l_ngx_family_tmp_tl 1867, 1903, 1909, 1910, 1915, 1921, 1922, 1927, 1933,
1934
__ngx_gloss_description: 4I, 598, 600, 605, 622
\g_ngx_gloss_link_color_str 634, 661, 772
\l_ngx_glossary_columns_int 2I, 688, 976, 979, 985
\l_ngx_glossary_style_str 579, 681, 1007

\l__lngx_glosses_page_number_bool	692, 945
\l__lngx_gls_bold_bool	705, 938, 1029
\l__lngx_gls_expansion_case_str	578, 672, 857, 891, 1039, 1079, 1128
\l__lngx_gls_section_number_bool	701, 1113
\l__lngx_gls_sectioning_str	697, 1109, 1112
\l__lngx_gls_sorting_order_str	577, 665, 996
\g__lngx_gls_use_order_seq	581, 820, 823, 995
\g__lngx_ipa_main_features_extra_tl	1290, 1295, 1296, 1303
\g__lngx_ipa_main_font_features_tl	1219, 1549
\g__lngx_ipa_main_font_tl	1408, 1550
\g__lngx_ipa_main_fonts_prop	1219, 1301
\g__lngx_ipa_mono_font_features_tl	1307, 1557
\g__lngx_ipa_mono_font_tl	1408, 1558
\g__lngx_ipa_mono_fonts_prop	1307
\g__lngx_ipa_sans_font_features_tl	1307, 1553
\g__lngx_ipa_sans_font_tl	1408, 1554
\g__lngx_ipa_sans_fonts_prop	1307
\g__lngx_math_bold_features_tl	353, 500, 550
\g__lngx_math_bold_font_tl	364, 551
\g__lngx_math_bold_fonts_prop	353, 499
\g__lngx_math_features_tl	353, 492, 545
\g__lngx_math_font_features_tl	473
\g__lngx_math_font_tl	362, 546
\g__lngx_math_fonts_prop	353, 491
\l__lngx_nfss_tmp_tl	1868, 2121, 2124, 2131, 2133
\l__lngx_normalfont_tmp_tl	1865
\g__lngx_old_style_user_choice_bool	135, 508
\g__lngx_page_ref_int	46, 575, 792, 794, 796, 803, 814
\l__lngx_selectfont_tmp_tl	1866, 1892, 1897, 1898
\l__lngx_separator_str	580, 583, 713, 841, 1163
\l__lngx_separator_tl	709
\g__lngx_text_main_features_extra_tl	224, 229, 230, 237
\g__lngx_text_main_font_features_tl	153, 473, 534
\g__lngx_text_main_font_tl	342, 535
\g__lngx_text_main_fonts_prop	153, 235
\g__lngx_text_mono_font_features_tl	241, 473, 542
\g__lngx_text_mono_font_tl	342, 543
\g__lngx_text_mono_fonts_prop	241
\g__lngx_text_sans_font_features_tl	241, 473, 538
\g__lngx_text_sans_font_tl	342, 539
\g__lngx_text_sans_fonts_prop	241
__lngx_tmp_text:	589, 591
\lngxipa	11, 17, 21, 1209, 1214, 1506
\lngxlogo	16, 1817, 1833, 1838, 1849
\lngxpkg	1830

\loadlanguages	15, 1631
\localecounter	1689
logical	15, 1685

M

\MakeLinkTarget	1026, 1072, 1125
\MakeLinkTarget*	51, 54
math	7, 353
math _{bold}	7, 353
math _{bold} _{features}	7, 353
math _{features}	7, 353
mode commands:	
\mode_leave_vertical:	616, 946
msg commands:	
\msg_info:nnn	86, 91
\msg_new:nnn	59, 1723
\msg_warning:nnn	1746, 1763
\multicol	40

N

native _{numbering}	15, 1696
newcm	6, 370
newcm _{mono}	6, 392
newcm _{regular}	6, 403
newcm _{regular} _{mono}	6, 425
newcm _{regular} _{sans}	6, 414
newcm _{sans}	6, 381
\NewCommandCopy	126
\NewDocumentCommand	742, 832, 837, 1161, 1206, 1818
\NewDocumentEnvironment	974
\newfontfamily	1787
\newgloss	8, 20, 745
\NewTaggingSocket	613
\NewTaggingSocketPlug	615
no _{bold}	10, 705
\noindent	982
\normalfont	2152, 2218

O

off	15, 1692
\ogLaTeX	5, 125
old _{style} _{numbers}	6, 135
old _{style} _{one}	6, 135

P

page _{numbers}	10, 692
\par	722
\parfillskip	934

\parindent	937
pdfannot commands:	
\pdfannot_dict_put:nnn	590
\pdfstringdef	589
prg commands:	
\prg_do_nothing:	605, 933, 934, 935, 936, 937, 960, 967
\prg_new_conditional:Nnn	1973, 1991, 2008, 2025, 2042, 2059, 2066, 2081, 2101
\prg_return_false:	1982, 2000, 2017, 2034, 2051, 2072, 2087, 2107
\prg_return_true:	1980, 1998, 2015, 2032, 2049, 2070, 2085, 2105
prop commands:	
\prop_gclear_new:N	154, 242, 245, 354, 357, 1220, 1308, 1311
\prop_gput:Nnn	186, 211, 235, 284, 312, 335, 1252, 1277, 1301, 1350, 1378, 1401
\prop_map_inline:Nn	482, 491, 499, 1521
\ProvideDocumentCommand	1107
\providelanguage	15, 1586
\ProvidesExplPackage	2, 36, 51, 98, 556, 1173, 1562, 1776, 1857

Q

\quad	947, 966
-------------	----------

R

\raisebox	130, 132
\ref	46
\relax	129, 130, 131, 132
\RenewDocumentCommand	128
\renewgloss	8, 47, 831
\RequirePackage	11, 14, 17, 20, 23, 26, 29, 32, 107, 113, 118, 123, 564, 1185, 1190, 1195, 1199, 1203, 1572, 1576, 1580, 1743, 1760, 1784
\rightskip	933
\rmdefault	1877
\roman	1713

S

\section	43
section_number	10, 701
sectioning	10, 697
\selectfont	2165, 2231
separator	10, 709
seq commands:	
\seq_clear:N	848, 994, 1120
\seq_gclear_new:N	581, 753
\seq_gput_right:Nn	809, 823
\seq_if_empty:NTF	883, 1068
\seq_if_in:NnTF	806, 820
\seq_map_inline:Nn	884, 1069, 1118, 1121
\seq_pop_left:NN	850, 1015
\seq_put_right:Nn	1122
\seq_remove_duplicates:N	46

\seq_set_eq:NN	995
\seq_set_from_clist:Nn	849
\seq_sort:Nn	998
\seq_use:Nn	1152
\l_tmpa_seq	848, 849, 850, 883, 884, 994, 995, 998, 1015, 1068, 1069, 1118
\l_tmpb_seq	1120, 1122, 1152
\setfontfamily	1477, 1484, 1491
\setmainfont	444
\setmathfont	455, 462
\setmonofont	450
\setsansfont	447
\setupglossing	9, 741
\sfdefault	1878
\smallskip	931
socket commands:	
\socket_assign_plug:nn	608, 639, 779, 1701
\socket_if_exist:nTF	586, 638, 776
\socket_new:nn	597, 1677
\socket_new_plug:nnn	587, 599, 603, 1679, 1686, 1693
\socket_use:n	611, 1704
\softernormalfont	19, 2248
\softersuperfontfamily	19, 2183
\softestnormalfont	19, 2258
\softestsuperfontfamily	19, 2194
\softnormalfont	19, 2206
\softsuperfontfamily	19, 2138
sort	9, 665
sort commands:	
\sort_return_same:	1002
\sort_return_swapped:	1000
str commands:	
\c_colon_str	1010, 1165
\str_case:nn	857, 891, 996, 1039, 1079, 1128, 2154, 2220
\str_clear:N	169, 180, 194, 195, 267, 278, 295, 296, 630, 748, 886, 1014, 1235, 1246, 1260, 1261, 1333, 1344, 1361, 1362
\str_clear_new:N	576, 577, 578, 579, 580
\str_compare:nNnTF	999
\str_if_eq:nnTF	1007, 1109
\str_lowercase:n	45, 749, 852, 888
\str_replace_all:Nnn	171, 185, 201, 202, 269, 283, 302, 303, 1237, 1251, 1267, 1268, 1335, 1349, 1368, 1369
\str_set:Nn	170, 181, 196, 200, 268, 279, 297, 301, 583, 631, 726, 749, 841, 851, 887, 1016, 1163, 1236, 1247, 1262, 1266, 1334, 1345, 1363, 1367
\str_set_eq:NN	667, 676, 683
\str_use:N	188, 213, 286, 314, 713, 751, 754, 757, 760, 765, 807, 810, 821, 824, 861, 868, 875, 881, 896, 904, 912, 919, 1027, 1044, 1053, 1062, 1112, 1254, 1279, 1352, 1380, 1702
\l_tmpa_str	169, 170, 171, 173, 180, 181, 185, 188, 194, 196, 201, 213, 267, 268, 269, 271, 278, 279, 283, 286, 295, 297, 302, 314, 630, 631, 748, 749, 751, 754,

757, 760, 765, 807, 810, 821, 824, 851, 861, 868, 875, 881, 886, 887, 896, 904,
912, 919, 1014, 1016, 1027, 1044, 1053, 1062, 1235, 1236, 1237, 1239, 1246, 1247,
1251, 1254, 1260, 1262, 1267, 1279, 1333, 1334, 1335, 1337, 1344, 1345, 1349,
1352, 1361, 1363, 1368, 1380
\l_tmpb_str 195, 200, 202, 204, 207, 215, 296, 301, 303, 305, 308, 316, 1261,
1266, 1268, 1270, 1273, 1281, 1362, 1367, 1369, 1371, 1374, 1382
strict 15, 1678
style 9, 681
\superfontfamily 18, 2129
sys commands:
\sys_if_engine_luatex:TF 71, 110, 1182

T

tag commands:
\tag_mc_begin:n 625, 653, 954, 964, 1022
\tag_mc_end: 617, 651, 949, 962, 1018, 1036
\tag_struct_begin:n 619, 950, 1019
\tag_struct_end: 652, 963, 1037
T_{EX} and L^AT_EX 2_ε commands:
\@currentmetafamily 1885
\@encoding 1892, 1903, 1915, 1927
\glw@gl 83
\texorpdfstring 1832, 1848
text_bold 11, 153
text_bold_features 11, 153
text_bold_italic 12, 153
text_bold_italic_features 12, 153
text_bold_slanted 12, 153
text_bold_slanted_features 12, 153
text_bold_swash 12, 153
text_bold_swash_features 12, 153
text commands:
\text_lowercase:n 859, 894, 1042, 1082, 1131
\text_titlecase_all:n 182, 197, 280, 298, 866, 902, 1051, 1089, 1138, 1248,
1263, 1346
\text_titlecase_first:n 873, 910, 1060, 1096, 1145
text_italic 12, 153
text_italic_features 12, 153
text_main_extra_features 13, 223
text_main_font 11, 342
text_mono_bold 12, 241
text_mono_bold_features 12, 241
text_mono_bold_italic 12, 241
text_mono_bold_italic_features 12, 241
text_mono_bold_slanted 12, 241
text_mono_bold_slanted_features 13, 241
text_mono_bold_swash 13, 241

<code>text_mono_bold_swash_features</code>	I3, 24I
<code>text_mono_extra_features</code>	I3, 24I
<code>text_mono_font</code>	I2, 342
<code>text_mono_italic</code>	I2, 24I
<code>text_mono_italic_features</code>	I2, 24I
<code>text_mono_slanted</code>	I2, 24I
<code>text_mono_slanted_features</code>	I2, 24I
<code>text_mono_small_caps</code>	I3, 24I
<code>text_mono_small_caps_features</code>	I3, 24I
<code>text_mono_swash</code>	I3, 24I
<code>text_mono_swash_features</code>	I3, 24I
<code>text_mono_upright</code>	I2, 24I
<code>text_mono_upright_features</code>	I2, 24I
<code>text_sans_bold</code>	I2, 24I
<code>text_sans_bold_features</code>	I2, 24I
<code>text_sans_bold_italic</code>	I2, 24I
<code>text_sans_bold_italic_features</code>	I2, 24I
<code>text_sans_bold_slanted</code>	I2, 24I
<code>text_sans_bold_slanted_features</code>	I2, 24I
<code>text_sans_bold_swash</code>	I2, 24I
<code>text_sans_bold_swash_features</code>	I2, 24I
<code>text_sans_extra_features</code>	I3, 24I
<code>text_sans_font</code>	I2, 342
<code>text_sans_italic</code>	I2, 24I
<code>text_sans_italic_features</code>	I2, 24I
<code>text_sans_slanted</code>	I2, 24I
<code>text_sans_slanted_features</code>	I2, 24I
<code>text_sans_small_caps</code>	I2, 24I
<code>text_sans_small_caps_features</code>	I2, 24I
<code>text_sans_swash</code>	I2, 24I
<code>text_sans_swash_features</code>	I2, 24I
<code>text_sans_upright</code>	I2, 24I
<code>text_sans_upright_features</code>	I2, 24I
<code>text_slanted</code>	I2, I53
<code>text_slanted_features</code>	I2, I53
<code>text_small_caps</code>	I2, I53
<code>text_small_caps_features</code>	I2, I53
<code>text_swash</code>	I2, I53
<code>text_swash_features</code>	I2, I53
<code>text_upright</code>	II, I53
<code>text_upright_features</code>	II, I53
<code>\textbf</code>	938, I029, I073
<code>\textit</code>	I720
<code>\textsc</code>	9, I30, 735

<code>\thechapter</code>	1640
<code>\theenumii</code>	1711, 1712
<code>\theenumiii</code>	1713, 1714
<code>\theenumiv</code>	1715, 1716
<code>\theequation</code>	1673
<code>\thefigure</code>	1661
<code>\thefootnote</code>	1667
<code>\thempfootnote</code>	1670
<code>\thepage</code>	1658
<code>\theparagraph</code>	1652
<code>\thesection</code>	1643
<code>\thesubparagraph</code>	1655
<code>\thesubsection</code>	1646
<code>\thesubsubsection</code>	1649
<code>\thetable</code>	1664
tl commands:	
<code>\c_space_tl</code>	843, 1010, 1165
<code>\tl_clear:N</code>	799, 847, 1013, 1750, 1751, 1767, 1768, 1898, 1910, 1922, 1934, 2133
<code>\tl_clear_new:N</code>	568, 569, 570
<code>\tl_const:Nn</code>	1877, 1878, 1879
<code>\tl_gclear_new:N</code>	155, 172, 224, 243, 246, 252, 270, 355, 358, 750, 1221, 1238, 1290, 1309, 1312, 1318, 1336, 2120
<code>\tl_gput_right:Nn</code>	203, 229, 304, 327, 483, 492, 500, 1269, 1295, 1370, 1393, 1522
<code>\tl_gset:Nn</code>	756, 2123
<code>\tl_if_empty:NTF</code>	206, 230, 307, 330, 1272, 1296, 1373, 1396, 2146
<code>\tl_if_eq:NnTF</code>	1979, 1997, 2014, 2031, 2048, 2069, 2084, 2104
<code>\tl_new:N</code>	1583, 1865, 1866, 1867, 1868, 1870, 1871, 1872, 1873, 1874
<code>\tl_set:Nn</code>	711, 800, 1737, 1755, 1880, 1881, 1884, 1887, 1888, 1892, 1902, 1903, 1907, 1914, 1915, 1919, 1926, 1927, 1931, 1938, 1942, 1946, 1950, 1954, 1958, 1962, 1966, 1970, 2131, 2140, 2208
<code>\tl_set_eq:NN</code>	1736, 1896, 1908, 1920, 1932
<code>\tl_use:N</code>	853, 860, 867, 874, 893, 895, 901, 903, 909, 911, 918, 942, 1031, 1032, 1043, 1052, 1071, 1076, 1682, 2148, 2215, 2270, 2273, 2276, 2279, 2282
<code>\tl_use:n</code>	1061
<code>\l_tmpa_tl</code>	799, 800, 808, 847, 850, 853, 1013, 1015, 1016, 1031, 1735, 1736, 1741, 1742, 1744, 1748, 1750, 1754, 1756, 1758, 1761, 1765, 1767
<code>\l_tmpb_tl</code>	1737, 1741, 1751, 1755, 1756, 1768
<code>\ttdefault</code>	1879

U

<code>\umgla</code>	5, 82
use commands:	
<code>\use:N</code>	143, 802, 813, 881, 919, 1111, 2160, 2168, 2173, 2226, 2234, 2239
<code>\use:n</code>	1110
<code>\use_ii:nnnnn</code>	801, 812
<code>\UseTaggingSocket</code>	763