

Parallel typesetting for critical editions: the **eledpar** package*

Peter Wilson
Herries Press[†]
Maïeul Rouquette[‡]

Abstract

The **eledmac** package, which is based on the PLAIN T_EX set of EDMAC macros, has been used for some time for typesetting critical editions. The **eledpar** package is an extension to **eledmac** which enables texts and their critical apparatus to be typeset in parallel, either in two columns or on pairs of facing pages.

Note that before September 2012, **eledpar** was called **ledpar**. The changes from **ledmac**/**ledpar** to **eledmac**/**eledpar** is explained in **ledmac** documentation.

eledpar provides many tools and options. Normally, they are all documented in this file. Also provided is a help folder, “examples”. The folder contains additional examples (although not for all cases).

To report bugs, please go to **ledmac**’s GitHub page and click “New Issue”: <https://github.com/maieul/ledmac/issues/>. You must open an account with github.com to access my page (maieul/ledmac). GitHub accounts are free for open-source users. You can report bug in English or in French (better).

You can subscribe to the **eledmac** email list in:
<http://geekographie.maieul.net/146>

Contents

1 Introduction	4
2 The eledpar package	4
2.1 General	5
3 Parallel columns	5

*This file (**eledpar.dtx**) has version number v1.14.0, last revised 2015/03/22.

[†]herries dot press at earthlink dot net

[‡]maieul at maieul dot net

4 Facing pages	7
4.1 Basic usage	7
4.2 Text width	7
4.3 Setting	7
4.4 Critical and familiar footnotes	8
4.4.1 Note size setting	8
4.4.2 Notes for one side only	8
4.4.3 Familiar notes called in the right side, but to be printed in the left side	8
5 Left and right texts	9
6 Numbering text lines and paragraphs	10
7 Verse	12
8 Side notes	14
9 Parallel ledgroups	14
9.1 Parallel ledgroups and <code>setspace</code> package	15
10 Sectioning commands	15
11 Implementation overview	16
12 Preliminaries	16
12.1 Messages	17
13 Sectioning commands	18
14 Line counting	20
14.1 Choosing the system of lineation	20
14.2 Line-number counters and lists	23
14.3 Reading the line-list file	24
14.4 Commands within the line-list file	25
14.5 Writing to the line-list file	33
15 Marking text for notes	36
15.1 Specific hooks and commands for notes	36
15.1.1 Create commands to memorize display options	37
15.1.2 Boolean flags for notes to be printed in one side only	37
15.1.3 Familiar footnotes without marks	37
15.1.4 Create hooks	38
15.1.5 Init standards series (A,B,C,D,E,Z)	38
16 Pstart numbers dumping and restoration	38
17 Parallel environments	39

18 Paragraph decomposition and reassembly	41
18.1 Boxes, counters, <code>\pstart</code> and <code>\pend</code>	42
18.2 Processing one line	46
18.3 Line and page number computation	49
18.4 Line number printing	52
18.5 Pstart number printing in side	54
18.6 Add insertions to the vertical list	55
18.7 Penalties	56
18.8 Printing leftover notes	57
19 Footnotes	57
19.1 Normal footnote formatting	57
19.2 Footnotes output specific to <code>\Pages</code>	58
20 Cross referencing	62
21 Side notes	62
22 Familiar footnotes	64
23 Verse	65
24 Naming macros	67
25 Counts and boxes for parallel texts	67
26 Fixing babel	69
27 Parallel columns	72
28 Parallel pages	77
29 Sections' titles' commands	89
30 Page break/no page break, depending on the specific line	90
31 Parallel ledgroup	90
32 The End	93
Appendix A Some things to do when changing version	94
Appendix A.1 Migration to <code>eledpar</code> 1.4.3	94
References	94
Index	94
Change History	105

1 Introduction

The EDMAC macros [LW90] for typesetting critical editions of texts have been available for use with TeX for some years. Since EDMAC became available there had been a small but constant demand for a version of EDMAC that could be used with L^AT_EX. The *eledmac* package was introduced in 2003 in an attempt to satisfy that request.

Some critical editions contain texts in more than one form, such as a set of verses in one language and their translations in another. In such cases there is a desire to be able to typeset the two texts, together with any critical apparatus, in parallel. The *eledpar* package is an extension to *eledmac* that enables two texts and their apparatus to be set in parallel, either in two columns or on pairs of facing pages.

The package has to try and coerce T_EX into paths it was not designed for. Use of the package, therefore, may produce some surprising results.

This manual contains a general description of how to use *eledpar* starting in section 2; the complete source code for the package, with extensive documentation (in sections 11 through 32); and an Index to the source code. As *eledpar* is an adjunct to *eledmac* I assume that you have read the *eledmac* manual. Also *eledpar* requires *eledmac* to be used, preferably at least version 0.10 (2011/08/22). You do not need to read the source code for this package in order to use it but doing so may help to answer any questions you might have. On a first reading, I suggest that you should skip anything after the general documentation in sections 2 until 11, unless you are particularly interested in the innards of *eledpar*.

2 The *eledpar* package

A file may mix *numbered* and *unnumbered* text. Numbered text is printed with marginal line numbers and can include footnotes and endnotes that are referenced to those line numbers: this is how you'll want to print the text that you're editing. Unnumbered text is not printed with line numbers, and you can't use *eledmac*'s note commands with it: this is appropriate for introductions and other material added by the editor around the edited text.

The *eledpar* package lets you typeset two *numbered* texts in parallel. This can be done either as setting the 'Leftside' and 'Rightside' texts in two columns or on facing pages. In the paired pages case footnotes are placed at the bottom of the page on which they are called out — that is, footnotes belonging to the left are set at the foot of a left (even numbered) page, and those for right texts are at the bottom of the relevant right (odd numbered) page. However, in the columnar case, all footnotes are set at the bottom left of the page on which they are called out — they are not set below the relevant column. The line numbering schemes need not be the same for the two texts.

2.1 General

`eledmac` essentially puts each chunk of numbered text (the text within a `\pstart` ... `\pend`) into a box and then following the `\pend` extracts the text line by line from the box to number and print it. More precisely, the text is first put into the box as though it was being typeset as normal onto a page and any notes are stored without being typeset. Then each typeset line is extracted from the box and any notes for that line are recalled. The line, with any notes, is then output for printing, possibly with a line number attached. Effectively, all the text is typeset and then afterwards all the notes are typeset.

`eledpar` similarly puts the left and right chunks into boxes but can't immediately output the text after a `\pend` — it has to wait until after both the left and right texts have been collected before it can start processing. This means that several boxes are required and possibly TeX has to store a lot of text in its memory; both the number of potential boxes and memory are limited. If TeX's memory is overfilled the recourse is to reduce the amount of text stored before printing.

`\maxchunks` It is possible to have multiple chunks in the left and right texts before printing them. The macro `\maxchunks{<num>}` specifies the maximum number of chunks within the left or right texts. This is initially set as:

```
\maxchunks{5120}
```

meaning that there can be up to 5120 chunks in the left text and up to 5120 chunks in the right text, requiring a total of 10240 boxes. If you need more chunks then you can increase `\maxchunks`. The `\maxchunks` must be called in the preamble.

TeX has a limited number of boxes; if you get an error message along the lines of 'no room for a new box', then load the package `etex`, which needs `pdflatex` or `xelatex`. If you `\maxchunks` is too little you can get a `eledmac` error message along the lines: 'Too many `\pstart` without printing. Some text will be lost.' then you will have to either increase `\maxchunks` or use the parallel printing commands (`\Columns` or `\Pages`) more frequently.

When typesetting verse using `\syntax`, each line is treated as a chunk, so be warned that if you are setting parallel verses you might have to increase `\maxchunks` much more than it appears at first sight.

In general, `eledmac` is a TeX resource hog, and `eledpar` only makes things worse in this respect.

3 Parallel columns

`pairs` Numbered text that is to be set in columns must be within a `pairs` environment. Within the environment the text for the lefthand and righthand columns is placed within the `Leftside` and `Rightside` environments, respectively; these are described in more detail below in section 5.

`\Columns` The command `\Columns` typesets the texts in the previous pair of `Leftside` and `Rightside` environments. The general scheme for parallel columns looks like this:

```
\begin{pairs}
```

```

\begin{Leftside} ... \end{Leftside}
\begin{Rightside} ... \end{Rightside}
\Columns
\begin{Leftside} ... \end{Leftside}
...
\end{pairs}
\Columns

```

Keep in mind that the `\Columns` **must be** outside of the `pairs` environment. You can use the macro `\AtBeginPairs` to insert a code at the beginning of each `pairs` environments. That could be useful to add the `\sloppy` macro to prevent overfull hboxes in two columns.

```
\AtBeginPairs{\sloppy}
```

There is no required pagebreak before or after the columns.

`\Lcolwidth` The lengths `\Lcolwidth` and `\Rcolwidth` are the widths of the left and right columns, respectively. By default, these are:

```
\setlength{\Lcolwidth}{0.45\textwidth}
```

```
\setlength{\Rcolwidth}{0.45\textwidth}
```

They may be adjusted if one text tends to be ‘bulkier’ than the other.

`\columnrulewidth` The macro `\columnseparator` is called between each left/right pair of lines. By default it inserts a vertical rule of width `\columnrulewidth`. As this is initially defined to be 0pt the rule is invisible. For a visible rule between the columns you could try:

```
\setlength{\columnrulewidth}{0.4pt}
```

You can also modify `\columnseparator` if you want more control.

`\columnspan` By default, columns are positioned to the right of the page. However, you use `\columnspan{L}` to align them to the left, or `\columnspan{C}` to center them.

When you use `\stanza`, the visible rule may shift when a verse has a hanging indent. To prevent shifting, use `\setstanzaindent` outside the `Leftside` or `Rightside` environment.

`\beforecolumnseparator` By default, the spaces around column separator are the same as the space:

`\aftercolumnseparator`

- On the left of columns, if columns are aligned right.
- On the right of columns, if columns are aligned left.
- On both the Left and Right columns, if columns are centered.

You can redefine `\beforecolumnseparator` and `\aftercolumnseparator` length to define spaces before or after the column separator, instead of letting `eledpar` calculate them automatically.

```
\setlength{\beforecolumnseparator}{length}
```

```
\setlength{\aftercolumnseparator}{length}
```

`\widthliketwocolumns` If you want to revert to the previous behavior, just set with a negative value. If you want to mix two-column with single-column text, you can align horizontally single-column text to two-column text with `\widthliketwocolumnstrue`. To reset this feature, use `\widthliketwocolumnsfalse`. You can also call `\widthliketwocolumns` as a global option when loading `eledmac` or `eledpar`.

`\Xnoteswidthliketwocolumns` In most cases, you should use `\widthliketwocolumns` in combination with `\Xnoteswidthliketwocolumns` and `\notesXwidthliketwocolumns` to align the critical/familiar footnotes with the two columns. See `eledmac`'s handbook for more details.

4 Facing pages

4.1 Basic usage

`pages` Numbered text that is to be set on facing pages must be within a `pages` environment. Within the environment the text for the lefthand and righthand pages is placed within the `Leftside` and `Rightside` environments, respectively.

`\Pages` The command `\Pages` typesets the texts in the previous pair of `Leftside` and `Rightside` environments. The general scheme for parallel pages looks like this:

```
\begin{pages}
\begin{Leftside} ... \end{Leftside}
\begin{Rightside} ... \end{Rightside}
\begin{Leftside} ... \end{Leftside}
...
\end{pages}
\Pages
```

The `Leftside` text is set on lefthand (even numbered) pages and the `Rightside` text is set on righthand (odd numbered) pages. Each `\Pages` command starts a new even numbered page. After parallel typesetting is finished, a new page is started. Note that the `\Pages` **must** be outside of the `pages` environment.

4.2 Text width

`\Lcolwidth` Within the `pages` environment the lengths `\Lcolwidth` and `\Rcolwidth` are the widths of the left and right pages, respectively. By default, these are set to the normal `textwidth` for the document, but can be changed within the environment if necessary.

`\Rcolwidth`

4.3 Setting

`\goalfraction` When doing parallel pages `eledpar` has to guess where TeX is going to put page-breaks and hopefully get there first in order to put the pair of texts on their proper pages. When it thinks that the fraction `\goalfraction` of a page has been filled, it finishes that page and starts on the other side's text. The definition is:

```
\newcommand*{\goalfraction}{0.9}
```

If you think you can get more on a page, increase this. On the other hand, if some left text overflows onto an odd numbered page or some right text onto an even page, try reducing it, for instance by:

```
\renewcommand*{\goalfraction}{0.8}
```

4.4 Critical and familiar footnotes

Of course, in “Facing pages”, the `eledmac` both critical and familiar footnotes can be used. However, some specific points must be taken into consideration.

4.4.1 Note size setting

Since `eledpar` v.1.13.0, long notes in facing pages can flow from left to right pages, and *vice-versa*. However, the `eledmac` default setting for the maximum allotted size to notes is greater than `\textheight`. That makes impossible for long notes to flow across pages.¹ We have not changed this default setting, because we don’t want to break compatibility with older version of `eledmac`. So, you **MUST** change the default setting via `\maxhXnotes` (for critical notes) `\maxhnotesX` (for familiar notes). Both commands are explained in handbook (5.4.9 p. 26). As and advisable setting:

```
\maxhXnotes{0.6\textheight}
\maxhnotesX{0.6\textheight}
```

4.4.2 Notes for one side only

<code>\onlyXside</code> <code>\onlysideX</code>	<p>You may want to typeset notes on one side only (either left or right). Use <code>\onlyXside[⟨s⟩]{⟨p⟩}</code> to set critical notes, and <code>\onlysideX[⟨s⟩]{⟨p⟩}</code> to set familiar notes. <code>{⟨p⟩}</code> must be set to L for notes to be confined only on the left side and to R for notes to be confined only on the right side.</p>
--	--

4.4.3 Familiar notes called in the right side, but to be printed in the left side

<code>\footnoteXnomk</code> <code>\footnoteXmk</code>	<p>As often happens, the left side has less room for text. We may want to call familiar notes in the right side while using at the same time the available space in the left side to print them.</p>
--	--

To achieve this, we call `\footnoteXnomk{⟨notecontent⟩}` in the left side. X is to be replaced by the series letter. We do this call in the left side after the word which matches up to the one in the right side after which we want to insert the actual footnote mark.

In the right side, we call `\footnoteXmk` at the place we want to have the footnote mark. X is to be replaced by the series letter. For example:

¹The same applies to L^AT_EX normal notes. Read <http://tex.stackexchange.com/a/228283/7712> for technical informations.


```

\begin{Leftside}
\beginnumbering
\pstart
  A little cat\footnoteAnomk{A note.}. And so one ...
\pend
\endnumbering
\end{Leftside}
\begin{Rightside}
\beginnumbering
\pstart
  Un petit chat\footnotemk. And so one ...
\pend
\endnumbering
\end{Rightside}

```

5 Left and right texts

Parallel texts are divided into Leftside and Rightside. The form of the contents of these two are independent of whether they will be set in columns or pages.

Leftside The left text is put within the **Leftside** environment and the right text likewise in the **Rightside** environment. The number of **Leftside** and **Rightside** environments must be the same.

Within these environments you can designate the line numbering scheme(s) to be used. The **eledmac** package originally used counters for specifying the numbering scheme; now both **eledmac** and the **eledpar** package use macros instead. Following `\firstlinenum{<num>}` the first line number will be *<num>*, and following `\linenumincrement{<num>}` only every *<num>*th line will have a printed number. Using these macros inside the **Leftside** and **Rightside** environments gives you independent control over the left and right numbering schemes. The `\firstsublinenum` and `\sublinenumincrement` macros correspondingly set the numbering scheme for sublines. The starred versions change both left and right numbering schemes.

\pstart In a serial (non-parallel) mode, each numbered paragraph, or chunk, is contained between the `\pstart` and `\pend` macros, and the paragraph is output when the `\pend` macro occurs. The situation is somewhat different with parallel typesetting as the left text (contained within `\pstart` and `\pend` groups within the **Leftside** environment) has to be set in parallel with the right text (contained within its own `\pstart` and `\pend` groups within the corresponding **Rightside** environment) the `\pend` macros cannot immediately initiate any typesetting — this has to be controlled by the `\Columns` or `\Pages` macros. Several chunks may be specified within a **Leftside** or **Rightside** environment. A multi-chunk text then looks like:

```

\begin{...side}
% \beginnumbering
\pstart first chunk \pend

```

```

\pstart  second chunk \pend
...
\pstart  last chunk \pend
% \endnumbering
\end{...side}

```

Numbering, via `\beginnumbering` and `\endnumbering`, may extend across several `Leftside` or `Rightside` environments. Remember, though, that the left/right sides are effectively independent of each other.

Generally speaking, controls like `\firstlinenum` or `\linenummargin` apply to sequential and left texts. To effect right texts only they have to be within a `Rightside` environment. `\lineationR` macro is the equivalent of `eledmac \lineation` macro for the right side. `\lineation*` macro is the equivalent of `eledmac \lineation` macro for both sides. If you are using the `babel` package with different languages (via, say, `\selectlanguage`) for the left and right texts it is particularly important to select the appropriate language within the `Leftside` and `Rightside` environments. The initial language selected for the right text is the `babel` package's default. Also, it is the *last* `\selectlanguage` in a side that controls the language used in any notes for that side when they get printed. If you are using multilingual notes then it is probably safest to explicitly specify the language(s) for each note rather than relying on the language selection for the side. The right side language is also applied to the right side line numbers.

Corresponding left and right sides must have the same number of paragraph chunks — if there are four on the left there must be four on the right, even if some are empty. The start of each pair of left and right chunks are aligned horizontally on the page. The ends may come at different positions — if one chunk is shorter than the other then blank lines are output on the shorter side until the end of the longer chunk is reached.

However, sometime if the left pstarts are much greater than right pstarts, or *vice-versa*, you can decide to shift the pstarts on the left and right side. That means the start of pstarts are not aligned horizontally on the page, the shift is offset at the end of each double pages. To enable this function, load `eledpar` with the option `shiftedpstarts`.

6 Numbering text lines and paragraphs

`\beginnumbering` Each section of numbered text must be preceded by `\beginnumbering` and followed by `\endnumbering`, like:

```

\beginnumbering
<text>
\endnumbering

```

These have to be separately specified within `Leftside` and `Rightside` environments.

The `\beginnumbering` macro resets the line number to zero, reads an auxiliary file called `<jobname>.nn` (where `<jobname>` is the name of the main input file for

this job, and `nn` is 1 for the first numbered section, 2 for the second section, and so on), and then creates a new version of this auxiliary file to collect information during this run. Separate auxiliary files are maintained for right hand texts and these are named `\jobname.nnR`, using the ‘R’ to distinguish them from the left hand and serial (non-parallel) texts.

`\memorydump` The command `\memorydump` effectively performs an `\endumbering` immediately followed by a `\beginnumbering` while not restarting the numbering sequence. This has the effect of clearing TeX’s memory of previous texts and any associated notes, allowing longer apparent streams of parallel texts. The command should be applied to both left and right texts, and after making sure that all previous notes have been output. For example, along the lines of:

```
\begin{Leftside}
  \beginnumbering
  ...
\end{Leftside}
\begin{Rightside}
  \beginnumbering
  ...
\end{Rightside}
\Pages
\begin{Leftside}
  \memorydump
  ...
\end{Leftside}
\begin{Rightside}
  \memorydump
  ...
```

`\Rlineflag` The value of `\Rlineflag` is appended to the line numbers of the right texts. Its default definition is:

```
\newcommand*\Rlineflag{\R}
```

This may be useful for parallel columns but for parallel pages it might be more appropriate to redefine it as:

`\printlinesR` `\renewcommand*\Rlineflag{\}`. The `\printlines` macro is ordinarily used to print the line number references for critical footnotes. For footnotes from right side texts a special version is supplied, called `\printlinesR`, which incorporates `\Rlineflag`. (The macro `\ledsavedprintlines` is a copy of the original `\printlines`, just in case ...). As provided, the package makes no use of `\printlinesR` but you may find it useful. For example, if you only use the B footnote series in righthand texts then you may wish to flag any line numbers in those footnotes with the value of `\Rlineflag`. You could do this by putting the following code in your preamble:

```
\let\oldBfootfmt\Bfootfmt
\renewcommand{\Bfootfmt}[3]{%
  \let\printlines\printlinesR
  \oldBfootfmt{#1}{#2}{#3}}
```

It's possible to insert a number at every `\pstart` command. You must use the `\numberpstarttrue` command to have it. You can stop the numerotation with `\numberpstartfalse`. You can redefine the commands `\thepstartL` and `\thepstartR` to change style. The numbering restarts on each `\beginnumbering`

7 Verse

If you are typesetting verse with `eledmac` you can use the `\stanza` construct, and you can also use this in right or left parallel texts. In this case each verse line is a chunk which has two implications. (1) you can unexpectedly exceed the `\maxchunks` limit or the overall limit on the number of boxes, and (2) left and right verse lines are matched, which may not be desirable if one side requires more print lines for verse lines than the other does.

`astanza` `eledpar` provides an `astanza` environment which you can use instead of `\stanza` (simply replace `\stanza` by `\begin{astanza}` and add `\end{astanza}` after the ending `\&`). Within the `astanza` environment each verse line is treated as a paragraph, so there must be no blank lines in the environment otherwise there will be some extraneous vertical spacing.

If you get an error message along the lines of ‘Missing number, treated as zero `\sza@00`’ it is because you have forgotten to use `\setstanzaindents` to set the stanza indents.

`\skipnumbering` The command `\skipnumbering` when inserted in a line of parallel text causes the numbering of that particular line to be skipped. This can useful if you are putting some kind of marker (even if it is only a blank line) between stanzas. Remember, parallel texts must be numbered and this provides a way to slip in an ‘unnumbered’ line.

The `astanza` environment forms a chunk but you may want to have more than one stanza within the chunk. Here are a couple of ways of doing that with a blank line between each internal stanza, and with each stanza numbered. First some preliminary definitions:

```
\newcommand*{\stanzanum}[2][\stanzaindentbase]{%
  \hspace{-#1\llap{\textbf{#2}}}\hspace{#1\ignorespaces}
\newcommand{\interstanza}{\par\mbox{}\skipnumbering}
```

And now for two stanzas in one. In this first example the line numbering repeats for each stanza.

```
\setstanzaindents{1,0,1,0,1,0,1,0,1,0,1}
\begin{pairs}
\begin{Leftside}
  \firstlinenum{2}
  \linenumincrement{1}
  \beginnumbering
  \begin{astanza}
```

```

\stanzanum{1} First in first stanza &
                Second in first stanza &
                Second in first stanza &
                Third in first stanza &
                Fourth in first stanza &
\interstanza
\setline{2}\stanzanum{2} First in second stanza &
                Second in second stanza &
                Second in second stanza &
                Third in second stanza &
                Fourth in second stanza \&
\end{astanza}
...

```

And here is a slightly different way of doing the same thing, but with the line numbering being continuous.

```

\setstanzaindents{1,0,1,0,1,0,0,1,0,1,0,1}
\begin{pairs}
\begin{Leftside}
\firstlinenum{2}
\linenumincrement{1}
\beginnumbering
\begin{astanza}
\stanzanum{1} First in first stanza &
                Second in first stanza &
                Second in first stanza &
                Third in first stanza &
                Fourth in first stanza &
\strut &
\stanzanum{2}\advanceline{-1} First in second stanza &
                Second in second stanza &
                Second in second stanza &
                Third in second stanza &
                Fourth in second stanza \&
\end{astanza}
...

```

`\hangingsymbol` Like in `eledmac`, you could redefine the command `\hangingsymbol` to insert a character in each hanging line. If you use it, you must run \LaTeX two time. Example for the French typography

```
\renewcommand{\hangingsymbol}{[,]}
```

You can also use it to force hanging verse to be flush right:

```
\renewcommand{\hangingsymbol}{\protect\hfill}
```

When you use `\lednopb` make sure to use it on both sides in the corresponding verses to keep the pages in sync.

8 Side notes

As in `eledmac`, you must use one of the following commands to add side notes: `\ledsidenote`, `\ledleftnote`, `\ledrightnote`, `\ledouterote`, `\ledinnerrote`.

The `\sidenotemargin` defines the margin of the sidenote for either left or right side, depending on the current environment. You can use `\sidenotemargin*` to define it for both sides.

9 Parallel ledgroups

You can also make parallel ledgroups (see the documentation of `eledmac` about ledgroups). To do it you have:

- To load `eledpar` package with the `parledgroup` option, or to add `\parledgrouptrue`.
- To push each ledgroup between `\pstart... \pend` command.

See the following example:

```
\begin{pages}
\begin{Leftside}
\beginnumbering
\pstart
\begin{ledgroup}
ledgroup content
\end{ledgroup}
\pend
\pstart
\begin{ledgroup}
ledgroup content
\end{ledgroup}
\pend
\endnumbering
\end{Leftside}
\begin{Rightside}
\beginnumbering
\pstart
\begin{ledgroup}
ledgroup content
\end{ledgroup}
\pend
\pstart
\begin{ledgroup}
ledgroup content
\end{ledgroup}
\pend
\endnumbering
\end{Rightside}
\Pages
```

```
\end{pages}
```

You can add sectioning a sectioning command, following this scheme:

```
\begin{..side}
  \beginnumbering
  \pstart
  \section{First ledgroup title}
\pend
  \pstart
  \begin{ledgroup}\skipnumbering
    ledgroup content
  \end{ledgroup}
\pend
  \pstart
  \section{Second ledgroup title}
\pend
  \pstart
  \begin{ledgroup}\skipnumbering
    ledgroup content
  \end{ledgroup}
\pend
\endnumbering
\end{..side}
```

9.1 Parallel ledgroups and setspace package

If you use the `setspace` package and want your notes in parallel ledgroups to be single-spaced (not half-spaced or double-spaced), just add to your preamble:

```
\let\parledgroupnotespacing\singlespacing
```

In effect, to have correct spacing, don't change the font size of your notes.

10 Sectioning commands

`\eledsectnotoc`

The standard sectioning commands of `eledmac` are available, and provide parallel sectionings, for both two-column and two-page layout. By default, the section commands of the right side are not added to the table of contents. But you can change it, using `\eledsectnotoc{<arg>}`, where `<arg>` could be `L` (for left side) or `R` (for right side).

`\eledsectmark`

By default, the `LATEX` marks for header are token from left side. You can change it, using `\eledsectmark{<arg>}`, where `<arg>` could be `L` (for left side) or `R` (for right side).

11 Implementation overview

TeX is designed to process a single stream of text, which may include footnotes, tables, and so on. It just keeps converting its input into a stream typeset pages. It was not designed for typesetting two texts in parallel, where it has to alternate from one to the other. Further, TeX essentially processes its input one paragraph at a time — it is very difficult to get at the ‘internals’ of a paragraph such as the individual lines in case you want to number them or put some mark at the start or end of the lines.

`eledmac` solves the problem of line numbering by putting the paragraph in typeset form into a box, and then extracting the lines one by one from the box for TeX to put them onto the page with the appropriate page breaks. Most of the `eledmac` code is concerned with handling this box and its contents.

`eledpar`’s solution to the problem of parallel texts is to put the two texts into separate boxes, and then appropriately extract the pairs of lines from the boxes. This involves duplicating much of the original box code for an extra right text box. The other, smaller, part of the code is concerned with coordinating the line extractions from the boxes.

The package code is presented in roughly in the same order as in `eledmac`.

12 Preliminaries

Announce the name and version of the package, which is targetted for LaTeX2e. The package also requires the `eledmac` package.

```
1 (*code)
2 \NeedsTeXFormat{LaTeX2e}
3 \ProvidesPackage{eledpar}[2015/03/22 v1.14.0 eledmac extension for parallel texts]%
4
```

Few commands use `\xspace` command.

```
5 \RequirePackage{xspace}%
```

With the option ‘`shiftedpstarts`’ a long `pstart` on the left side (or in the right side) doesn’t make a blank on the corresponding `pstart`, but the blank is put on the bottom of the page. Consequently, the `pstarts` on the parallel pages are shifted, but the shift stops at every end of pages. The `\shiftedverses` is kept for backward compatibility.

```
\ifshiftedpstarts
```

```
6 \newif\ifshiftedpstarts
7 \let\shiftedversestrue\shiftedpstartstrue
8 \let\shiftedversesfalse\shiftedpstartsfalse
9 \DeclareOption{shiftedverses}{\shiftedpstartstrue}
10 \DeclareOption{shiftedpstarts}{\shiftedpstartstrue}
11 \DeclareOption{parledgroup}{\parledgrouptrue}
```

`\ifwidthliketwocolumns` The `\widthliketwocolumns` option can be called both in `eledpar` and `eledmac`.


```
12 \DeclareOption{widthliketwocolumns}{\widthliketwocolumnstrue}%
```

```
13 \ProcessOptions%
```

As noted above, much of the code is a duplication of the original `eledmac` code to handle the extra box(es) for the right hand side text, and sometimes for the left hand side as well. In order to distinguish I use ‘R’ or ‘L’ in the names of macros for the right and left code. The specifics of ‘L’ and ‘R’ are normally hidden from the user by letting the `Leftside` and `Rightside` environments set things up appropriately.

```
\ifl@dpairing \ifl@dpairing is set TRUE if we are processing parallel texts and \ifl@dpaging
\ifl@dpaging is also set TRUE if we are doing parallel pages. \ifledRcol is set TRUE if we
\ifledRcol are doing the right hand text. They are defined in eledmac.
```

```
\Lcolwidth The widths of the left and right parallel columns (or pages).
```

```
\Rcolwidth 14 \newdimen\Lcolwidth
15 \Lcolwidth=0.45\textwidth
16 \newdimen\Rcolwidth
17 \Rcolwidth=0.45\textwidth
18
```

12.1 Messages

All the error and warning messages are collected here as macros.

```
\eledpar@error
```

```
19 \newcommand{\eledpar@error}[2]{\PackageError{eledpar}{#1}{#2}}
```

```
\led@err@TooManyPstarts
```

```
20 \newcommand*{\led@err@TooManyPstarts}{%
21 \eledpar@error{Too many \string\pstart\space without printing.
22 Some text will be lost}{\@ehc}}
```

```
\led@err@BadLeftRightPstarts
```

```
23 \newcommand*{\led@err@BadLeftRightPstarts}[2]{%
24 \eledpar@error{The numbers of left (#1) and right (#2)
25 \string\pstart s do not match}{\@ehc}}
```

```
\led@err@LeftOnRightPage
```

```
\led@err@RightOnLeftPage 26 \newcommand*{\led@err@LeftOnRightPage}{%
27 \eledpar@error{The left page has ended on a right page}{\@ehc}}
28 \newcommand*{\led@err@RightOnLeftPage}{%
29 \eledpar@error{The right page has ended on a left page}{\@ehc}}
```

13 Sectioning commands

`\section@numR` This is the right side equivalent of `\section@num`.

Each section will read and write an associated ‘line-list file’, containing information used to do the numbering. Normally the file will be called `\jobname\section@num`, where `num` is the section number. However, for right side texts the file is called `\jobname\section@numR`. The `\extensionchars` applies to the right side files just as it does to the normal files.

```
30 \newcount\section@numR
31 \section@numR=\z@
```

`\ifpst@rtedL` `\ifpst@rtedL` is set FALSE at the start of left side numbering, and similarly for `\ifpst@rtedR`. `\ifpst@rtedL` is defined in `eledmac`.

```
32 \pst@rtedLfalse
33 \newif\ifpst@rtedR
34
```

`\beginnumberingR` This is the right text equivalent of `\beginnumbering`, and begins a section of numbered text.

```
35 \newcommand*\beginnumberingR{%
36   \ifnumberingR
37     \led@err@NumberingStarted
38     \endnumberingR
39   \fi
40   \global\l@dnumpstartsR \z@
41   \global\pst@rtedRfalse
42   \global\numberingRtrue
43   \global\advance\section@numR \@ne
44   \global\absline@numR \z@
45   \gdef\normal@page@breakR{}
46   \gdef\l@prev@pbR{}
47   \gdef\l@prev@nopbR{}
48   \global\line@numR \z@
49   \global\@lockR \z@
50   \global\sub@lockR \z@
51   \global\sublines@false
52   \global\let\next@page@numR\relax
53   \global\let\sub@change\relax
54   \message{Section \the\section@numR R }%
55   \line@list@stuffR{\jobname.\extensionchars\the\section@numR R}%
56   \l@dend@stuff
57   \setcounter{pstartR}{1}
58   \begingroup
59   \initnumbering@sectcountR
60   \gdef\eled@sectionsR@{}%
61   \if@noeled@sec\else%
62     \makeatletter\inputIfFileExists{\jobname.eledsec\the\section@numR R}{-}{\makeatother}%
63     \immediate\openout\eled@sectioningR@out=\jobname.eledsec\the\section@numR R\relax%
```

```
64 \fi%
65 }
```

`\endnumbering` This is the left text version of the regular `\endnumbering` and must follow the last text for a left text numbered section. It sets `\ifpst@rtedL` to FALSE. It is fully defined in `eledmac`.

`\endnumberingR` This is the right text equivalent of `\endnumbering` and must follow the last text for a right text numbered section.

```
66 \def\endnumberingR{%
67 \ifnumberingR
68 \global\numberingRfalse
69 \normal@pars
70 \ifl@dpairing
71 \global\pst@rtedRfalse
72 \else
73 \ifx\insertlines@listR\empty\else
74 \global\noteschanged@true
75 \fi
76 \ifx\line@listR\empty\else
77 \global\noteschanged@true
78 \fi
79 \fi
80 \ifnoteschanged@
81 \led@mess@NotesChanged
82 \fi
83 \else
84 \led@err@NumberingNotStarted
85 \fi
86 \endgroup
87 \if@noeled@sec\else%
88 \immediate\closeout\eled@sectioningR@out%
89 \fi%
90 }
91
```

`\initnumbering@sectcountR` We don't want the numbering of the right-side section commands to be continuous with the numbering of the left side, we switch the \LaTeX counter in `\numberingR`.

```
92 \newcounter{chapterR}
93 \newcounter{sectionR}
94 \newcounter{subsectionR}
95 \newcounter{subsubsectionR}
96 \newcommand{\initnumbering@sectcountR}{
97 \let\c@chapter\c@chapterR
98 \let\c@section\c@sectionR
99 \let\c@subsection\c@subsectionR
100 \let\c@subsubsection\c@subsubsectionR
101 }
```

`\pausenumberingR` These are the right text equivalents of `\pausenumbering` and `\resumenumbering`.
`\resumenumberingR`

```

102 \newcommand*{\pausenumberingR}{%
103   \endnumberingR\global\numberingRtrue}
104 \newcommand*{\resumenumberingR}{%
105   \ifnumberingR
106     \global\pst@rtedRtrue
107     \global\advance\section@numR \@ne
108     \led@mess@SectionContinued{\the\section@numR R}%
109     \line@list@stuffR{\jobname.\extensionchars\the\section@numR R}%
110     \l@dend@stuff
111     \begingroup%
112     \initnumbering@sectcountR%
113   \else
114     \led@err@numberingShouldHaveStarted
115     \endnumberingR
116     \beginnumberingR
117   \fi}
118
```

`\memorydumpL` `\memorydump` is a shorthand for `\pausenumbering\resumenumbering`. This will
`\memorydumpR` clear the memorised stuff for the previous chunks while keeping the numbering
going.

```

119 \newcommand*{\memorydumpL}{%
120   \endnumbering
121   \numberingtrue
122   \global\pst@rtedLtrue
123   \global\advance\section@num \@ne
124   \led@mess@SectionContinued{\the\section@num}%
125   \line@list@stuff{\jobname.\extensionchars\the\section@num}%
126   \l@dend@stuff}
127 \newcommand*{\memorydumpR}{%
128   \endnumberingR
129   \numberingRtrue
130   \global\pst@rtedRtrue
131   \global\advance\section@numR \@ne
132   \led@mess@SectionContinued{\the\section@numR R}%
133   \line@list@stuffR{\jobname.\extensionchars\the\section@numR R}%
134   \l@dend@stuff}
135
```

14 Line counting

14.1 Choosing the system of lineation

Sometimes you want line numbers that start at 1 at the top of each page; sometimes you want line numbers that start at 1 at each `\pstart`; other times you want line

numbers that start at 1 at the start of each section and increase regardless of page breaks. `eledpar` lets you choose different schemes for the left and right texts.

`\ifbypstart@R` The `\ifbypage@R` and `\ifbypstart@R` flag specify the current lineation system:

`\bypstart@Rtrue` • line-of-page : `bypstart@R = false` and `bypage@R = true`.

`\bypstart@Rfalse` • line-of-pstart : `bypstart@R = true` and `bypage@R = false`.

`\ifbypage@R` `\bypage@Rtrue` `\bypage@Rfalse` `eledpar` will use the line-of-section system unless instructed otherwise.

```

136 \newif\ifbypage@R
137 \newif\ifbypstart@R

```

`\lineationR` `\lineationR{<word>}` is the macro used to select the lineation system for right texts. Its argument is a string: either `page`, `pstart` or `section`.

```

138 \newcommand*{\lineationR}[1]{%
139   \ifnumbering
140     \led@err@LineationInNumbered
141   \else
142     \def\@tempa{#1}\def\@tempb{page}%
143     \ifx\@tempa\@tempb
144       \global\bypage@Rtrue
145       \global\bypstart@Rfalse
146     \else
147       \def\@tempb{pstart}%
148       \ifx\@tempa\@tempb
149         \global\bypage@Rfalse
150         \global\bypstart@Rtrue
151       \else
152         \def\@tempb{section}
153         \ifx\@tempa\@tempb
154           \global\bypage@Rfalse
155           \global\bypstart@Rfalse
156         \else
157           \led@warn@BadLineation
158         \fi
159       \fi
160     \fi
161   \fi}}

```

`\lineation*` `\lineation*` change the lineation system for the side.

```

162 \WithSuffix\newcommand\lineation*[1]{%
163   \lineation{#1}%
164   \lineationR{#1}%
165 }%

```

`\linenummargin` You call `\linenummargin{<word>}` to specify which margin you want your right text's line numbers in; it takes one argument, a string. You can put the line numbers in the same margin on every page using `left` or `right`; or you can use `inner` or `outer` to get them in the inner or outer margins. You can change this

within a numbered section, but the change may not take effect just when you'd like; if it's done between paragraphs nothing surprising should happen.

For right texts the selection is recorded in the count `\line@marginR`, otherwise in the count `\line@margin`: 0 for left, 1 for right, 2 for outer, and 3 for inner.

```

166 \newcount\line@marginR
167 \renewcommand*{\linenummargin}[1]{%
168   \l@getline@margin{#1}%
169   \ifnum\l@dttempcntb>\m@ne
170     \ifledRcol
171       \global\line@marginR=\l@dttempcntb
172     \else
173       \global\line@margin=\l@dttempcntb
174     \fi
175   \fi}}

```

By default put right text numbers at the right.

```

176 \line@marginR=\@ne
177

```

`\c@firstlinenumR` The following counters tell `eledmac` which right text lines should be printed with
`\c@linenumincrementR` line numbers. `firstlinenum` is the number of the first line in each section that gets a number; `linenumincrement` is the difference between successive numbered lines. The initial values of these counters produce labels on lines 5, 10, 15, etc. `linenumincrement` must be at least 1.

```

178 \newcounter{firstlinenumR}
179 \setcounter{firstlinenumR}{5}
180 \newcounter{linenumincrementR}
181 \setcounter{linenumincrementR}{5}

```

`\c@firstsublinenumR` The following parameters are just like `firstlinenumR` and `linenumincrementR`,
`\c@sublinenumincrementR` but for sub-line numbers. `sublinenumincrementR` must be at least 1.

```

182 \newcounter{firstsublinenumR}
183 \setcounter{firstsublinenumR}{5}
184 \newcounter{sublinenumincrementR}
185 \setcounter{sublinenumincrementR}{5}
186

```

`\firstlinenum` These are the user's macros for changing (sub) line numbers. They are defined in
`\linenumincrement` `eledmac v0.7`, but just in case I have started by `\provideing` them. The starred
`\firstsublinenum` versions are specific to `eledpar`.
`\sublinenumincrement` 187 `\providecommand*{\firstlinenum}{}`
`\firstlinenum*` 188 `\providecommand*{\linenumincrement}{}`
`\linenumincrement*` 189 `\providecommand*{\firstsublinenum}{}`
`\firstsublinenum*` 190 `\providecommand*{\sublinenumincrement}{}`
`\sublinenumincrement*` 191 `\renewcommand*{\firstlinenum}[1]{%`
 192 `\ifledRcol \setcounter{firstlinenumR}{#1}%`
 193 `\else \setcounter{firstlinenumR}{#1}%`
 194 `\fi}`

```

195 \renewcommand*{\linenumincrement}[1]{%
196   \ifledRcol \setcounter{linenumincrementR}{#1}%
197   \else      \setcounter{linenumincrement}{#1}%
198   \fi}
199 \renewcommand*{\firstsublinenum}[1]{%
200   \ifledRcol \setcounter{firstsublinenumR}{#1}%
201   \else      \setcounter{firstsublinenum}{#1}%
202   \fi}
203 \renewcommand*{\sublinenumincrement}[1]{%
204   \ifledRcol \setcounter{sublinenumincrementR}{#1}%
205   \else      \setcounter{sublinenumincrement}{#1}%
206   \fi}
207 \WithSuffix\newcommand\firstlinenum*[1]{\setcounter{firstlinenumR}{#1}\setcounter{firstlinenum}{#1}}
208 \WithSuffix\newcommand\linenumincrement*[1]{\setcounter{linenumincrementR}{#1}\setcounter{linenumincr
209 \WithSuffix\newcommand\firstsublinenum*[1]{\setcounter{subfirstlinenumR}{#1}\setcounter{subfirstlinenum
210 \WithSuffix\newcommand\sublinenumincrement*[1]{\setcounter{sublinenumincrementR}{#1}\setcounter{sublin

```

`\Rlineflag` This is appended to the line numbers of right text.

```

211 \newcommand*{\Rlineflag}{R}
212

```

`\linenumrepR` `\linenumrepR{<ctr>}` typesets the right line number `<ctr>`, and similarly `\sublinenumrepR`
`\sublinenumrepR` for subline numbers.

```

213 \newcommand*{\linenumrepR}[1]{\@arabic{#1}}
214 \newcommand*{\sublinenumrepR}[1]{\@arabic{#1}}
215

```

`\leftlinenumR` `\leftlinenumR` and `\rightlinenumR` are the macros that are called to print the
`\rightlinenumR` right text's marginal line numbers. Much of the code for these is common and is
`\l@dlinenumR` maintained in `\l@dlinenumR`.

```

216 \newcommand*{\leftlinenumR}{%
217   \l@dlinenumR
218   \kern\linenumsep}
219 \newcommand*{\rightlinenumR}{%
220   \kern\linenumsep
221   \l@dlinenumR}
222 \newcommand*{\l@dlinenumR}{%
223   \numlabfont\linenumrepR{\line@numR}\Rlineflag%
224   \ifsublines@
225     \ifnum\subline@num>\z@
226       \unskip\fullstop\sublinenumrepR{\subline@numR}%
227     \fi
228   \fi}
229

```

14.2 Line-number counters and lists

We need another set of counters and lists for the right text, corresponding to those in `eledmac` for regular or left text.

`\line@numR` The count `\line@numR` stores the line number that's used in the right text's marginal line numbering and in notes. The count `\subline@numR` stores a sub-line number that qualifies `\line@numR`. The count `\absline@numR` stores the absolute number of lines since the start of the right text section: that is, the number we've actually printed, no matter what numbers we attached to them.

```
230 \newcount\line@numR
231 \newcount\subline@numR
232 \newcount\absline@numR
233
```

`\line@listR` Now we can define the list macros that will be created from the line-list file. They are directly analagous to the left text ones. The full list of action codes and their meanings is given in the `eledmac` manual.

`\actions@listR` Here are the commands to create these lists:

```
234 \list@create{\line@listR}
235 \list@create{\insertlines@listR}
236 \list@create{\actionlines@listR}
237 \list@create{\actions@listR}
238 \list@create{\sw@listR}%
239 \list@create{\sw@list@inedtextR}%
240
```

`\linesinpar@listL` In order to synchronise left and right chunks in parallel processing we need to know how many lines are in each left and right text chunk, and the maximum of these for each pair of chunks.

`\linesinpar@listR`
`\maxlinesinpar@list`

```
241 \list@create{\linesinpar@listL}
242 \list@create{\linesinpar@listR}
243 \list@create{\maxlinesinpar@list}
244
```

`\page@numR` The right text page number.

```
245 \newcount\page@numR
246
```

14.3 Reading the line-list file

`\read@linelist` `\read@linelist{<file>}` is the control sequence that's called by `\beginnumbering` (via `\line@list@stuff`) to open and process a line-list file; its argument is the name of the file.

```
247 \renewcommand*{\read@linelist}[1]{%
```

We do do different things depending whether or not we are processing right text

```
248 \ifledRcol
249 \list@clear{\line@listR}%
250 \list@clear{\insertlines@listR}%
251 \list@clear{\actionlines@listR}%
252 \list@clear{\actions@listR}%
253 \list@clear{\linesinpar@listR}%
```



```

254 \list@clear{\linesonpage@listR}
255 \list@clear{\sw@listR}%
256 \list@clear{\sw@list@inedtextR}%
257 \else
258 \list@clearing@reg
259 \list@clear{\linesinpar@listL}%
260 \list@clear{\linesonpage@listL}%
261 \fi

```

Make sure that the `\maxlinesinpar@list` is empty (otherwise things will be thrown out of kilter if there is any old stuff still hanging in there).

```
262 \list@clear{\maxlinesinpar@list}
```

Now get the file and interpret it.

```

263 \get@linelistfile{#1}%
264 \endgroup

```

When the reading is done, we're all through with the line-list file. All the information we needed from it will now be encoded in our list macros. Finally, we initialize the `\next@actionline` and `\next@action` macros, which specify where and what the next action to be taken is.

```

265 \ifledRcol
266 \global\page@numR=\m@ne
267 \ifx\actionlines@listR\empty
268 \gdef\next@actionlineR{1000000}%
269 \else
270 \gl@p\actionlines@listR\to\next@actionlineR
271 \gl@p\actions@listR\to\next@actionR
272 \fi
273 \else
274 \global\page@num=\m@ne
275 \ifx\actionlines@list\empty
276 \gdef\next@actionline{1000000}%
277 \else
278 \gl@p\actionlines@list\to\next@actionline
279 \gl@p\actions@list\to\next@action
280 \fi
281 \fi}
282

```

This version of `\read@linelist` creates list macros containing data for the entire section, so they could get rather large. The `\memorydump` macro is available if you run into macro memory limitations.

14.4 Commands within the line-list file

This section defines the commands that can appear within a line-list file, except for `\@lab` which is in a later section among the cross-referencing commands it is associated with.

The macros with `action` in their names contain all the code that modifies the action-code list.

`\@nl@regR` `\@nl` does everything related to the start of a new line of numbered text. Exactly `\@nl` what it does depends on whether right text is being processed.

```

283 \newcommand{\@nl@regR}{%
284   \ifx\l@dchset@num\relax \else
285     \advance\absline@numR \@ne
286     \set@line@action
287     \let\l@dchset@num\relax
288     \advance\absline@numR \m@ne
289     \advance\line@numR \m@ne%    % do we need this?
290   \fi
291   \advance\absline@numR \@ne
292   \ifx\next@page@numR\relax \else
293     \page@action
294     \let\next@page@numR\relax
295   \fi
296   \ifx\sub@change\relax \else
297     \ifnum\sub@change>\z@
298       \sublines@true
299     \else
300       \sublines@false
301     \fi
302     \sub@action
303     \let\sub@change\relax
304   \fi
305   \ifcase\@lockR
306   \or
307     \@lockR \tw@
308   \or\or
309     \@lockR \z@
310   \fi
311   \ifcase\sub@lockR
312   \or
313     \sub@lockR \tw@
314   \or\or
315     \sub@lockR \z@
316   \fi
317   \ifsublines@
318     \ifnum\sub@lockR<\tw@
319       \advance\subline@numR \@ne
320     \fi
321   \else
322     \ifnum\@lockR<\tw@
323       \advance\line@numR \@ne \subline@numR \z@
324     \fi
325   \fi}
326
327 \renewcommand*{\@nl}[2]{%
```

```

328 \fix@page{#1}%
329 \ifledRcol
330   \@nl@regR
331 \else
332   \@nl@reg
333 \fi}
334

```

\last@page@numR We have to adjust \fix@page to handle parallel texts.

```

\fix@page 335 \newcount\last@page@numR
336   \last@page@numR=-10000
337 \renewcommand*{\fix@page}[1]{%
338   \ifledRcol
339     \ifnum #1=\last@page@numR
340     \else
341       \ifbypage@R
342         \line@numR \z@ \subline@numR \z@
343         \fi
344         \page@numR=#1\relax
345         \last@page@numR=#1\relax
346         \def\next@page@numR{#1}%
347       \fi
348     \else
349       \ifnum #1=\last@page@num
350       \else
351         \ifbypage@
352         \line@num \z@ \subline@num \z@
353         \fi
354         \page@num=#1\relax
355         \last@page@num=#1\relax
356         \def\next@page@num{#1}%
357         \listxadd{\normal@page@break}{\the\absline@num}
358       \fi
359     \fi}
360

```

\@adv The \@adv{<num>} macro advances the current visible line number by the amount specified as its argument. This is used to implement \advanceline.

```

361 \renewcommand*{\@adv}[1]{%
362   \ifsublines@
363     \ifledRcol
364       \advance\subline@numR by #1\relax
365       \ifnum\subline@numR<\z@
366         \led@warn@BadAdvancelineSubline
367         \subline@numR \z@
368       \fi
369     \else
370       \advance\subline@num by #1\relax
371       \ifnum\subline@num<\z@

```

```

372         \led@warn@BadAdvancelineSubline
373         \subline@num \z@
374     \fi
375 \fi
376 \else
377     \ifledRcol
378         \advance\line@numR by #1\relax
379         \ifnum\line@numR<\z@
380             \led@warn@BadAdvancelineLine
381             \line@numR \z@
382         \fi
383     \else
384         \advance\line@num by #1\relax
385         \ifnum\line@num<\z@
386             \led@warn@BadAdvancelineLine
387             \line@num \z@
388         \fi
389     \fi
390 \fi
391 \set@line@action}
392

```

\@set The `\@set{<num>}` macro sets the current visible line number to the value specified as its argument. This is used to implement `\setline`.

```

393 \renewcommand*{\@set}[1]{%
394     \ifledRcol
395         \ifsublines@
396             \subline@numR=#1\relax
397         \else
398             \line@numR=#1\relax
399         \fi
400     \set@line@action
401 \else
402     \ifsublines@
403         \subline@num=#1\relax
404     \else
405         \line@num=#1\relax
406     \fi
407     \set@line@action
408 \fi}
409

```

\l@d@set The `\l@d@set{<num>}` macro sets the line number for the next `\pstart...` to the value specified as its argument. This is used to implement `\setlinenum`.

\l@dchset@num `\l@dchset@num` is a flag to the `\@l` macro. If it is not `\relax` then a linenum change is to be done.

```

410 \renewcommand*{\l@d@set}[1]{%
411     \ifledRcol
412         \line@numR=#1\relax

```

```

413 \advance\line@numR \@ne
414 \def\l@dchset@num{#1}
415 \else
416 \line@num=#1\relax
417 \advance\line@num \@ne
418 \def\l@dchset@num{#1}
419 \fi}
420 \let\l@dchset@num\relax
421

```

`\page@action` `\page@action` adds an entry to the action-code list to change the page number.

```

422 \renewcommand*{\page@action}{%
423 \ifledRcol
424 \xright@appenditem{\the\absline@numR}\to\actionlines@listR
425 \xright@appenditem{\next@page@numR}\to\actions@listR
426 \else
427 \xright@appenditem{\the\absline@num}\to\actionlines@list
428 \xright@appenditem{\next@page@num}\to\actions@list
429 \fi}

```

`\set@line@action` `\set@line@action` adds an entry to the action-code list to change the visible line number.

```

430 \renewcommand*{\set@line@action}{%
431 \ifledRcol
432 \xright@appenditem{\the\absline@numR}\to\actionlines@listR
433 \ifsublines@
434 \l@dtempcnta=-\subline@numR
435 \else
436 \l@dtempcnta=-\line@numR
437 \fi
438 \advance\l@dtempcnta by -5000\relax
439 \xright@appenditem{\the\l@dtempcnta}\to\actions@listR
440 \else
441 \xright@appenditem{\the\absline@num}\to\actionlines@list
442 \ifsublines@
443 \l@dtempcnta=-\subline@num
444 \else
445 \l@dtempcnta=-\line@num
446 \fi
447 \advance\l@dtempcnta by -5000\relax
448 \xright@appenditem{\the\l@dtempcnta}\to\actions@list
449 \fi}
450

```

`\sub@action` `\sub@action` adds an entry to the action-code list to turn sub-lineation on or off, according to the current value of the `\ifsublines@` flag.

```

451 \renewcommand*{\sub@action}{%
452 \ifledRcol
453 \xright@appenditem{\the\absline@numR}\to\actionlines@listR

```

```

454 \ifsublines@
455 \xright@appenditem{-1001}\to\actions@listR
456 \else
457 \xright@appenditem{-1002}\to\actions@listR
458 \fi
459 \else
460 \xright@appenditem{\the\absline@num}\to\actionlines@list
461 \ifsublines@
462 \xright@appenditem{-1001}\to\actions@list
463 \else
464 \xright@appenditem{-1002}\to\actions@list
465 \fi
466 \fi}
467

```

`\do@lockon` `\lock@on` adds an entry to the action-code list to turn line number locking on.
`\do@lockonR` The current setting of the sub-lineation flag tells us whether this applies to line numbers or sub-line numbers.

```

468 \newcount\@lockR
469 \newcount\sub@lockR
470
471 \newcommand*{\do@lockonR}{%
472 \xright@appenditem{\the\absline@numR}\to\actionlines@listR
473 \ifsublines@
474 \xright@appenditem{-1005}\to\actions@listR
475 \ifnum\sub@lockR=\z@
476 \sub@lockR \@ne
477 \else
478 \ifnum\sub@lockR=\thr@@
479 \sub@lockR \@ne
480 \fi
481 \fi
482 \else
483 \xright@appenditem{-1003}\to\actions@listR
484 \ifnum\@lockR=\z@
485 \@lockR \@ne
486 \else
487 \ifnum\@lockR=\thr@@
488 \@lockR \@ne
489 \fi
490 \fi
491 \fi}
492
493 \renewcommand*{\do@lockon}{%
494 \ifx\next\lock@off
495 \global\let\lock@off=\skip@lockoff
496 \else
497 \ifledRcol
498 \do@lockonR
499 \else

```

```

500     \do@lockonL
501     \fi
502 \fi}

```

`\lock@off` `\lock@off` adds an entry to the action-code list to turn line number locking off.

```

\do@lockoff 503
\do@lockoffR 504
\skip@lockoff 505 \newcommand{\do@lockoffR}{%
506   \xright@appenditem{\the\absline@numR}\to\actionlines@listR
507   \ifsublines@
508     \xright@appenditem{-1006}\to\actions@listR
509     \ifnum\sub@lockR=\tw@
510       \sub@lockR \thr@@
511     \else
512       \sub@lockR \z@
513     \fi
514   \else
515     \xright@appenditem{-1004}\to\actions@listR
516     \ifnum\@lockR=\tw@
517       \@lockR \thr@@
518     \else
519       \@lockR \z@
520     \fi
521   \fi}
522
523 \renewcommand*{\do@lockoff}{%
524   \ifledRcol
525     \do@lockoffR
526   \else
527     \do@lockoffL
528   \fi}
529 \global\let\lock@off=\do@lockoff
530

```

`\n@num` This macro implements the `\skipnumbering` command. It uses a new action code, namely 1007.

```

531 \providecommand*\n@num{}
532 \renewcommand*\n@num{%
533   \ifledRcol
534     \xright@appenditem{\the\absline@numR}\to\actionlines@listR
535     \xright@appenditem{-1007}\to\actions@listR
536   \else
537     \n@num@reg
538   \fi}
539

```

`\@ref` `\@ref` marks the start of a passage, for creation of a footnote reference. It takes `\insert@countR` two arguments:

- #1, the number of entries to add to `\insertlines@list` for this reference. This value for right text, here and within `\edtext`, which computes it and writes it to the line-list file, will be stored in the count `\insert@countR`.

540 `\newcount\insert@countR`

- #2, a sequence of other line-list-file commands, executed to determine the ending line-number. (This may also include other `\@ref` commands, corresponding to uses of `\edtext` within the first argument of another instance of `\edtext`.)

The first thing `\@ref` itself does is to add the specified number of items to the `\insertlines@list` list.

```
541 \renewcommand*{\@ref}[2]{%
542   \ifledRcol
543   \global\insert@countR=#1\relax
544   \loop\ifnum\insert@countR>\z@
545     \xright@appenditem{\the\absline@numR}\to\insertlines@listR
546     \global\advance\insert@countR \m@ne
547   \repeat
```

Next, process the second argument to determine the page and line numbers for the end of this lemma. We temporarily equate `\@ref` to a different macro that just executes its argument, so that nested `\@ref` commands are just skipped this time. Some other macros need to be temporarily redefined to suppress their action.

```
548   \begingroup
549     \let\@ref=\dummy@ref
550     \let\@lopR\@gobble
551     \let\page@action=\relax
552     \let\sub@action=\relax
553     \let\set@line@action=\relax
554     \let\@lab=\relax
555     \let\@sw\@gobbletwo%
556     #2
557     \global\endpage@num=\page@numR
558     \global\endline@num=\line@numR
559     \global\endsubline@num=\subline@numR
560   \endgroup
```

Now store all the information about the location of the lemma's start and end in `\line@list`.

```
561   \xright@appenditem%
562     {\the\page@numR|\the\line@numR|%
563     \ifsublines@ \the\subline@numR \else 0\fi|%
564     \the\endpage@num|\the\endline@num|%
565     \ifsublines@ \the\endsubline@num \else 0\fi}\to\line@listR
```

Finally, execute the second argument of `\@ref` again, to perform for real all the commands within it.

566 #2


```

567 \else
    And when not in right text
568 \ref@reg{#1}{#2}%
569 \fi}

\@pend \@pend{<num>} adds its argument to the \linesinpar@listL list, and analagously
\@pendR for \@pendR. If needed, it resets line number. We start off with a \providecommand
just in case an older version of eledmac is being used which does not define these
macros.

570 \providecommand*\@pend{[1]}{}
571 \renewcommand*\@pend{[1]}{}
572 \ifbypstart@global\line@num=0\fi%
573 \xright@appenditem{#1}\to\linesinpar@listL}
574 \providecommand*\@pendR{[1]}{}
575 \renewcommand*\@pendR{[1]}{}
576 \ifbypstart@Rglobal\line@numR=0\fi
577 \xright@appenditem{#1}\to\linesinpar@listR}
578

\@lopL \@lopL{<num>} adds its argument to the \linesonpage@listL list, and analagously
\@lopR for \@lopR. We start off with a \providecommand just in case an older version of
eledmac is being used which does not define these macros.

579 \providecommand*\@lopL{[1]}{}
580 \renewcommand*\@lopL{[1]}{}
581 \xright@appenditem{#1}\to\linesonpage@listL}
582 \providecommand*\@lopR{[1]}{}
583 \renewcommand*\@lopR{[1]}{}
584 \xright@appenditem{#1}\to\linesonpage@listR}
585

```

14.5 Writing to the line-list file

We've now defined all the counters, lists, and commands involved in reading the line-list file at the start of a section. Now we'll cover the commands that eledmac uses within the text of a section to write commands out to the line-list.

`\linenum@outR` The file for right texts will be opened on output stream `\linenum@outR`.

```
586 \newwrite\linenum@outR
```

`\iffirst@linenum@out@R` Once any file is opened on this stream, we keep it open forever, or else switch to
`\first@linenum@out@Rtrue` another file that we keep open.

```

\first@linenum@out@Rfalse 587 \newif\iffirst@linenum@out@R
588 \first@linenum@out@Rtrue

```

`\line@list@stuffR` This is the right text version of the `\line@list@stuff{<file>}` macro. It is called by `\beginnumberingR` and performs all the line-list operations needed at the start of a section. Its argument is the name of the line-list file.

```

589 \newcommand*{\line@list@stuffR}[1]{%
590   \read@linelist{#1}%
591   \iffirst@linenum@out@R
592     \immediate\closeout\linenum@outR
593     \global\first@linenum@out@Rfalse
594     \immediate\openout\linenum@outR=#1
595   \else
596     \if@minipage%
597       \leavevmode%
598     \fi%
599     \closeout\linenum@outR%
600     \openout\linenum@outR=#1%
601   \fi}
602

```

`\new@lineL` The `\new@lineL` macro sends the `\@nl` command to the left text line-list file, to mark the start of a new text line.

```

603 \newcommand*{\new@lineL}{%
604   \write\linenum@out{\string\@nl[\the\c@page][\thepage]}}

```

`\new@lineR` The `\new@lineR` macro sends the `\@nl` command to the right text line-list file, to mark the start of a new text line.

```

605 \newcommand*{\new@lineR}{%
606   \write\linenum@outR{\string\@nl[\the\c@page][\thepage]}}

```

`\flag@start` We enclose a lemma marked by `\edtext` in `\flag@start` and `\flag@end`: these send the `\@ref` command to the line-list file.

`\startsub` `\startsub` and `\endsub` turn sub-lineation on and off, by writing appropriate instructions to the line-list file.

```

607 \renewcommand*{\startsub}{\dimen0\lastskip
608   \ifdim\dimen0>0pt \unskip \fi
609   \ifledRcol \write\linenum@outR{\string\sub@on}%
610   \else      \write\linenum@out{\string\sub@on}%
611   \fi
612   \ifdim\dimen0>0pt \hskip\dimen0 \fi}
613 \def\endsub{\dimen0\lastskip
614   \ifdim\dimen0>0pt \unskip \fi
615   \ifledRcol \write\linenum@outR{\string\sub@off}%
616   \else      \write\linenum@out{\string\sub@off}%
617   \fi
618   \ifdim\dimen0>0pt \hskip\dimen0 \fi}
619

```

`\advanceline` You can use `\advanceline{<num>}` in running text to advance the current visible line-number by a specified value, positive or negative.

```

620 \renewcommand*{\advanceline}[1]{%
621   \ifledRcol \write\linenum@outR{\string\@adv[#1]}%
622   \else      \write\linenum@out{\string\@adv[#1]}%
623   \fi}

```

`\setline` You can use `\setline{<num>}` in running text (i.e., within `\pstart... \pend`) to set the current visible line-number to a specified positive value.

```
624 \renewcommand*{\setline}[1]{%
625   \ifnum#1<\z@
626     \led@warn@BadSetline
627   \else
628     \ifledRcol \write\linenum@outR{\string\@set[#1]}%
629     \else      \write\linenum@out{\string\@set[#1]}%
630   \fi
631 \fi}
```

`\setlinenum` You can use `\setlinenum{<num>}` before a `\pstart` to set the visible line-number to a specified positive value. It writes a `\l@d@set` command to the line-list file.

```
632 \renewcommand*{\setlinenum}[1]{%
633   \ifnum#1<\z@
634     \led@warn@BadSetlinenum
635   \else
636     \ifledRcol \write\linenum@outR{\string\l@d@set[#1]}
637     \else      \write\linenum@out{\string\l@d@set[#1]} \fi
638   \fi}
639
```

`\startlock` You can use `\startlock` or `\endlock` in running text to start or end line number locking at the current line. They decide whether line numbers or sub-line numbers are affected, depending on the current state of the sub-lineation flags.

```
\endlock
640 \renewcommand*{\startlock}{%
641   \ifledRcol \write\linenum@outR{\string\lock@on}%
642   \else      \write\linenum@out{\string\lock@on}%
643   \fi}
644 \def\endlock{%
645   \ifledRcol \write\linenum@outR{\string\lock@off}%
646   \else      \write\linenum@out{\string\lock@off}%
647   \fi}
648
```

`\skipnumbering` In numbered text, `\skipnumbering` in a line will suspend the numbering for that particular line. That is, line numbers are unchanged and no line number will be printed.

```
649 \renewcommand*{\skipnumbering}{%
650   \ifledRcol \write\linenum@outR{\string\n@num}%
651             \advanceline{-1}%
652   \else
653     \skipnumbering@reg
654   \fi}
655
```

15 Marking text for notes

The `\edtext` (or `\critext`) macro is used to create all footnotes and endnotes, as well as to print the portion of the main text to which a given note or notes is keyed. The idea is to have that lemma appear only once in the `.tex` file: all instances of it in the main text and in the notes are copied from that one appearance.

`\critext` requires two arguments. At any point within numbered text, you use it by saying:

```
\critext{#1}#2/
```

Similarly `\edtext` requires the same two arguments but you use it by saying:

```
\edtext{#1}{#2}
```

```
\critext And similarly for \edtext.
\edtext
\set@line The \set@line macro is called by \edtext to put the line-reference field and font
specifier for the current block of text into \l@d@nums.

656 \renewcommand*{\set@line}{%
657   \ifledRcol
658     \ifx\line@listR\empty
659       \global\noteschanged@true
660       \xdef\l@d@nums{000|000|000|000|000|000|000|000|\edfont@info}%
661     \else
662       \gl@p\line@listR\to\@tempb
663       \xdef\l@d@nums{\@tempb|\edfont@info}%
664       \global\let\@tempb=\undefined
665     \fi
666   \else
667     \ifx\line@list\empty
668       \global\noteschanged@true
669       \xdef\l@d@nums{000|000|000|000|000|000|000|000|\edfont@info}%
670     \else
671       \gl@p\line@list\to\@tempb
672       \xdef\l@d@nums{\@tempb|\edfont@info}%
673       \global\let\@tempb=\undefined
674     \fi
675   \fi}
676
```

15.1 Specific hooks and commands for notes

The `\eledmac \newseries@` initializes commands which are linked to notes series. However, to keep `eledmac` as light as possible, it does not define commands which are specific to `eledpar`. This is what does `\newseries@eledpar`. The specific hooks are also defined here.

`\newseries@eledpar`

```
677 \newcommand{\newseries@eledpar}[1]{%
```

15.1.1 Create commands to memorize display options

```
678 \csgdef{onlysideX@#1}{}%
```

```
679 \csgdef{onlyXside@#1}{}%
```

15.1.2 Boolean flags for notes to be printed in one side only

`eledpar` allows notes to be printed on one side only. We need boolean flags, and set them to true when a note series is not printed in one side even though it contains something.

```
680 \global\newbool{keepforXside@#1}%
```

```
681 \global\newbool{keepforsideX@#1}%
```

15.1.3 Familiar footnotes without marks

The `\footnoteXnomk` commands are for notes which are printed in the left side, while they are called in the right side. Basically, they set first toggle `\nomark@` to true, then call the `\footnoteX`. and finally add the footnote counter in the footnote counter list.

So declare the list.

```
682 \expandafter\list@create\csname footnote#1@mk\endcsname%
```

Then, declare the `\footnoteXnomk` command.

```
683 \expandafter\newcommand\csname footnote#1nomk\endcsname[1]{%
```

First step: just call the normal `\footnoteX`, saying that we don't want to print the mark.

```
684 \toggletrue{nomk@}%
```

```
685 \csuse{footnote#1}{##1}%
```

```
686 \togglefalse{nomk@}%
```

Second, and last, step: store the footnote counter in the footnote counters list. We use some `\let`, because `\xright@appenditem` is difficult to use with `\expandafter`.

```
687 \letcs{\@tmp}{footnote#1@mk}%
```

```
688 \numdef\@tmpa{\csuse{c@footnote#1}}%
```

```
689 \global\xright@appenditem{\@tmpa}\to\@tmp%
```

```
690 \global\cslet{footnote#1@mk}{\@tmp}%
```

```
691 }%
```

Then, declare the command which inserts the footnotemark in the right side.

```
692 \expandafter\newcommand\csname footnote#1mk\endcsname{%
```

Get the first element of the footnote mark list. As `\gl@p` is difficult to use with dynamic name macro, we use `\let` commands.

```
693 \letcs{\@tmp}{footnote#1@mk}%
```

```
694 \gl@p\@tmp\to\@tmpa%
```

```
695 \global\cslet{footnote#1@mk}{\@tmp}%
```

Set the footnotecounter with it. For the sake of security, we make a backup of the previous value.

```

696      \letcs{\old@footnote}{c@footnote#1}%
697      \setcounter{footnote#1}{\@tmpa}%

Define the footnote mark and print it
698      \protected@csxdef{@thefnmark#1}{\csuse{thefootnote#1}}%
699      \csuse{@footnotemark#1}%

Restore previous footnote counter and finally add space.
700      \setcounter{footnote#1}{\old@footnote}%
701      \xspace%
702  }%

End of \newseries@eledpar.
703 }%
```

15.1.4 Create hooks

Read the eledmac code handbook about `\newhookcommand@series`. Here, we create hooks which are specific to eledpar.

```

704 \newhookcommand@series{onlysideX}%
705
706 \newhookcommand@series{onlyXside}%
707
```

15.1.5 Init standards series (A,B,C,D,E,Z)

`\init@series@eledpar` `\newseries@eledpar` is called by `\newseries@`. However, this command is called before eledpar is loaded. Thus, we need to initiate a specific series hook for eledpar.

```

708 \newcommand{\init@series@eledpar}{%
709   \def\do##1{\newseries@eledpar{##1}}%
710   \dolistloop{\@series}%
711 }%
712 \init@series@eledpar%
```

16 Pstart numbers dumping and restoration

While in eledmac the footnotes are inserted in the same time as the `\pstart` ... `\pend` are read, in eledpar they are inserted when the `\Columns` or `\Pages` commands are called. Consequently, if we do nothing, the value of the `PstartL` and `PstartR` counters are not the same in the main text and in the notes. To solve this problem, we dump the values in two list (one by side) when processing `\pstart` and restore these at each `\pstart` when calling `\Columns` or `\Pages`. We also dump and restore the value of the boolean `\ifnumberpstart`.

So, first step, creating the lists. Here, “pc” means “public counters”.

```

\list@pstartL@pc
\list@pstartR@pc 713 \list@create{\list@pstartL@pc}%
                  714 \list@create{\list@pstartR@pc}%

```

Two commands to dump current pstarts. We prefer two commands to one with argument indicating the side, because the commands are short, and so we save one test (or a `\csname` construction).

```

\dump@pstartL@pc
\dump@pstartR@pc 715 \def\dump@pstartL@pc{%
                  716 \xright@appenditem{\the\c@pstartL}\to\list@pstartL@pc%
                  717 \global\cslet{numberpstart@L\the\l@dnumpstartsL}{\ifnumberpstart}%
                  718 }%
                  719
                  720 \def\dump@pstartR@pc{%
                  721 \xright@appenditem{\the\c@pstartR}\to\list@pstartR@pc%
                  722 \global\cslet{numberpstart@R\the\l@dnumpstartsR}{\ifnumberpstart}%
                  723 }%
                  724

```

```

\restore@pstartL@pc And so, the commands to restore them
\restore@pstartR@pc 725 \def\restore@pstartL@pc{%
                    726 \ifx\list@pstartL@pc\empty\else%
                    727 \gl@p\list@pstartL@pc\to\@temp%
                    728 \global\c@pstartL=\@temp%
                    729 \fi%
                    730 }%
                    731 \def\restore@pstartR@pc{%
                    732 \ifx\list@pstartR@pc\empty\else%
                    733 \gl@p\list@pstartR@pc\to\@temp%
                    734 \global\c@pstartR=\@temp%
                    735 \fi%
                    736 }%

```

17 Parallel environments

The initial set up for parallel processing is deceptively simple.

`pairs` The `pairs` environment is for parallel columns and the `pages` environment for parallel pages.

```

chapterinpages 737 \newenvironment{pairs}{%}
                738 \l@dpairingtrue
                739 \l@dpagingfalse
                740 \initnumbering@ssectcmd
                741 \at@begin@pairs%
                742 }{%
                743 \l@dpairingfalse
                744 }
                745

```

`\AtBeginPairs` The `\AtBeginPairs` macro just define a `\at@begin@pairs` macro, called at the beginning of each `pairs` environments.

```
746 \newcommand{\AtBeginPairs}[1]{\xdef\at@begin@pairs{#1}}%
747 \def\at@begin@pairs{%
748
```

The `pages` environment additionally sets the ‘column’ widths to the `\textwidth` (as known at the time the package is called). In this environment, there are two text in parallel on 2 pages. To prevent chapters starting on a lefthand page, the `\chapter` command is redefined to not clear pages.

```
749 \newenvironment{pages}{%
750 \let\oldchapter\chapter
751 \let\chapter\chapterinpages
752 \l@dpairingtrue
753 \l@dpagingtrue
754 \initnumbering@sectcmd
755 \setlength{\Lcolwidth}{\textwidth}%
756 \setlength{\Rcolwidth}{\textwidth}%
757 }{%
758 \l@dpairingfalse
759 \l@dpagingfalse
760 \let\chapter\oldchapter
761 }
762 \newcommand{\chapterinpages}{\thispagestyle{plain}%
763 \global\@topnum\z@
764 \@afterindentfalse
765 \secdef\@chapter\@schapter}
766
```

`ifinstanzaL` These boolean tests are switched by the `\stanza` command, using either the left or right side.

```
767 \newif\ifinstanzaL
768 \newif\ifinstanzaR
```

`Leftside` Within the `pairs` and `pages` environments the left and right hand texts are within `Leftside` and `Rightside` environments, respectively. The `Leftside` environment is simple, indicating that right text is not within its purview and using some particular macros.

```
769 \newenvironment{Leftside}{%
770 \ledRcolfalse
771 \setcounter{pstartL}{1}
772 \let\pstart\pstartL
773 \let\thepstart\thepstartL
774 \let\pend\pendL
775 \let\memorydump\memorydumpL
776 \Leftsidehook
777 \let\old@startstanza\@startstanza
778 \def\@startstanza[##1]{\global\instanzaLtrue\old@startstanza[##1]}
```



```

779 }{
780   \Leftsidehookend}

\Leftsidehook   Hooks into the start and end of the Leftside and Rightside environments. These
\Leftsidehookend are initially empty.
\Rightsidehook  781 \newcommand*{\Leftsidehook}{}
\Rightsidehookend 782 \newcommand*{\Leftsidehookend}{}
                  783 \newcommand*{\Rightsidehook}{}
                  784 \newcommand*{\Rightsidehookend}{}
                  785

Rightside       The Rightside environment is only slightly more complicated than the Leftside.
                  Apart from indicating that right text is being provided it ensures that the right
                  right text code will be used.

786 \newenvironment{Rightside}{%
787   \ledRcoltrue
788   \let\beginnumbering\beginnumberingR
789   \let\endnumbering\endnumberingR
790   \let\pausenumbering\pausenumberingR
791   \let\resumenumbering\resumenumberingR
792   \let\memorydump\memorydumpR
793   \let\thepstart\thepstartR
794   \let\pstart\pstartR
795   \let\pend\pendR
796   \let\ledpb\ledpbR
797   \let\lednopb\lednopbR
798   \let\lineation\lineationR
799   \Rightsidehook
800   \let\old@startstanza\@startstanza
801   \def\@startstanza[##1]{\global\instanzaRtrue\old@startstanza[##1]}
802 }{%
803   \ledRcolfalse
804   \Rightsidehookend
805 }
806

```

18 Paragraph decomposition and reassembly

In order to be able to count the lines of text and affix line numbers, we add an extra stage of processing for each paragraph. We send the paragraph into a box register, rather than straight onto the vertical list, and when the paragraph ends we slice the paragraph into its component lines; to each line we add any notes or line numbers, add a command to write to the line-list, and then at last send the line to the vertical list. This section contains all the code for this processing.

18.1 Boxes, counters, \pstart and \pend

`\num@linesR` Here are numbers and flags that are used internally in the course of the paragraph decomposition.

`\one@lineR`

`\par@lineR` When we first form the paragraph, it goes into a box register, `\l@dLcolrawbox` or `\l@dRcolrawbox` for right text, instead of onto the current vertical list. The `\ifnumberedpar@` flag will be `true` while a paragraph is being processed in that way. `\num@lines(R)` will store the number of lines in the paragraph when it's complete. When we chop it up into lines, each line in turn goes into the `\one@line` or `\one@lineR` register, and `\par@line(R)` will be the number of that line within the paragraph.

```
807 \newcount\num@linesR
```

```
808 \newbox\one@lineR
```

```
809 \newcount\par@lineR
```

`\pstartL` `\pstart` starts the paragraph by clearing the `\inserts@list` list and other relevant variables, and then arranges for the subsequent text to go into the appropriate box. `\pstart` needs to appear at the start of every paragraph that's to be numbered.

`\pstartR`

Beware: everything that occurs between `\pstart` and `\pend` is happening within a group; definitions must be global if you want them to survive past the end of the paragraph.

We have to have specific left and right `\pstart` when parallel processing; among other things because of potential changes in the linewidth.

```
810
```

```
811 \newcounter{pstartL}
```

```
812 \renewcommand{\thepstartL}{\bfseries\@arabic\c@pstartL}. }
```

```
813 \newcounter{pstartR}
```

```
814 \renewcommand{\thepstartR}{\bfseries\@arabic\c@pstartR}. }
```

```
815
```

```
816 \newcommandx*\pstartL}[1][1]{%
```

```
817   \if@nbreak%
```

```
818     \let\oldnbreak\@nbreaktrue%
```

```
819   \else%
```

```
820     \let\oldnbreak\@nbreakfalse%
```

```
821   \fi%
```

```
822   \@nbreaktrue%
```

```
823   \ifluatex%
```

```
824     \xdef\l@luatextextdir@L{\the\luatextextdir}%
```

```
825     \xdef\l@luatexpardir@L{\the\luatexpardir}%
```

```
826     \xdef\l@luatexbodydir@L{\the\luatexbodydir}%
```

```
827   \fi%
```

```
828   \ifnumbering \else%
```

```
829     \led@err@PstartNotNumbered%
```

```
830     \beginnumbering%
```

```
831   \fi%
```

```
832   \ifnumberedpar%
```

```
833     \led@err@PstartInPstart%
```

```
834 \pend%
835 \fi%
```

If this is the first \pstart in a numbered section, clear any inserts and set \ifpst@rtedL to FALSE.

```
836 \ifpst@rtedL\else%
837 \list@clear{\inserts@list}%
838 \global\let\next@insert=\empty%
839 \global\pst@rtedLtrue%
840 \fi%
841 \begingroup\normal@pars%
```

When parallel processing we check that we haven't exceeded the maximum number of chunks. In any event we grab a box for the forthcoming text.

```
842 \global\advance\l@dnumpstartsL \one%
843 \ifnum\l@dnumpstartsL>\l@dc@maxchunks%
844 \led@err@TooManyPstarts%
845 \global\l@dnumpstartsL=\l@dc@maxchunks%
846 \fi%
847 \global\setnamebox{\l@dc@l@rawbox\the\l@dnumpstartsL}=\vbox\bgroup%
848 \ifautopar\else%
849 \ifnumberpstart%
850 \ifsidepstartnum%
851 \else%
852 \thepstartL%
853 \fi%
854 \fi%
855 \fi%
856 \hsize=\Lcolwidth%
857 \numberedpar@true%
858 \iflabeledpstart\protected@edef\@currentlabel%
859 {\p@pstartL\thepstartL}\fi%
```

Dump the optional arguments

```
860 \ifstrempy{#1}%
861 {\csgdef{before@pstartL@the\l@dnumpstartsL}{\at@every@pstart}}%
862 {\csgdef{before@pstartL@the\l@dnumpstartsL}{\noindent#1}}%
863 }

864 \newcommandx*{\pstartR}[1][1]{%
865 \if@nbreak%
866 \let\@oldnbreak\@nbreaktrue%
867 \else%
868 \let\@oldnbreak\@nbreakfalse%
869 \fi%
870 \@nbreaktrue%
871 \ifluatex%
872 \xdef\l@luatextextdir@R{\the\luatextextdir}%
873 \xdef\l@luatexpardir@R{\the\luatexpardir}%
874 \xdef\l@luatexbodydir@R{\the\luatexbodydir}%
875 \fi%
```

```

876 \ifnumberingR \else%
877   \led@err@PstartNotNumbered%
878   \beginnumberingR%
879 \fi%
880 \ifnumberedpar@%
881   \led@err@PstartInPstart%
882   \pendR%
883 \fi%
884 \ifpst@rtedR\else%
885   \list@clear{\inserts@listR}%
886   \global\let\next@insertR=\empty%
887   \global\pst@rtedRtrue%
888 \fi%
889 \begingroup\normal@pars%
890 \global\advance\l@dnumpsstartsR \@ne%
891 \ifnum\l@dnumpsstartsR>\l@dc@maxchunks%
892   \led@err@TooManyPstarts%
893   \global\l@dnumpsstartsR=\l@dc@maxchunks%
894 \fi%
895 \global\setnamebox{\l@dRcolrawbox\the\l@dnumpsstartsR}=\vbox\bgroup%
896   \ifautopar\else%
897     \ifnumberpstart%
898       \ifsidepstartnum\else%
899         \thepstartR%
900       \fi%
901     \fi%
902   \fi%
903 \hsize=\Rcolwidth%
904 \numberedpar@true%
905 \iflabelpstart\protected@edef\@currentlabel%
906   {\p@pstartR\thepstartR}\fi%
907 \ifstrempy{#1}%
908   {\csgdef{before@pstartR@the\l@dnumpsstartsR}{\at@every@pstart}}%
909   {\csgdef{before@pstartR@the\l@dnumpsstartsR}{\noindent#1}}%
910 }

```

`\pendL` `\pend` must be used to end a numbered paragraph. Again we need a version that knows about left parallel texts.

```

911 \newcommand*{\pendL}[1][1]{%
912   \ifnumbering \else%
913     \led@err@PendNotNumbered%
914   \fi%
915   \ifnumberedpar@ \else%
916     \led@err@PendNoPstart%
917   \fi%

```

We set all the usual interline penalties to zero and then immediately call `\endgraf` to end the paragraph; this ensures that there'll be no large interline penalties to prevent us from slicing the paragraph into pieces. These penalties revert to the values that you set when the group for the `\vbox` ends.

```

918 \l@dzeropenalties%
919 \endgraf\global\num@lines=\prevgraf\egroup%
920 \global\par@line=0%

  End the group that was begun in the \pstart.
921 \endgroup%
922 \ignorespaces%
923 \@oldnbreak%
924 \dump@pstartL@pc%
925 \ifnumberpstart%
926   \addtocounter{pstartL}{1}%
927 \fi
928 \parledgroup@beforenotes@save{L}%

  Dump content of the optional argument.
929 \ifstrepty{#1}%
930   {\csgdef{after@pendL@the\l@dnumpstartsL}{\at@every@pend}}%
931   {\csgdef{after@pendL@the\l@dnumpstartsL}{\noindent#1}}%
932 }

```

`\pendR` The version of `\pend` needed for right texts.

```

933 \newcommand*{\pendR}[1][1]{%
934   \ifnumberingR \else%
935     \led@err@PendNotNumbered%
936   \fi%
937   \ifnumberedpar@ \else%
938     \led@err@PendNoPstart%
939   \fi%
940   \l@dzeropenalties%
941   \endgraf\global\num@linesR=\prevgraf\egroup%
942   \global\par@lineR=0%
943   \endgroup%
944   \ignorespaces%
945   \@oldnbreak%
946   \dump@pstartR@pc%
947   \ifnumberpstart%
948     \addtocounter{pstartR}{1}%
949   \fi%
950   \parledgroup@beforenotes@save{R}%
951   \ifstrepty{#1}%
952     {\csgdef{after@pendR@the\l@dnumpstartsR}{\at@every@pend}}%
953     {\csgdef{after@pendR@the\l@dnumpstartsR}{\noindent#1}}%
954 }
955

```

`\ifprint@last@after@pendL` Two booleans set to true, when the time is to print the last optional argument of
`\ifprint@last@after@pendR` a `\pend`.

```

956 \newif\ifprint@last@after@pendL%
957 \newif\ifprint@last@after@pendR%

```

18.2 Processing one line

For parallel texts we have to be able to process left and right lines independently. For sequential text we happily use the original `\do@line`. Otherwise ...

```

\l@dleftbox  A line of left text will be put in the box \l@dleftbox, and analogously for a line
\l@drightbox of right text.
958 \newbox\l@dleftbox
959 \newbox\l@drightbox
960

\countLline  We need to know the number of lines processed.
\countRline  961 \newcount\countLline
962   \countLline \z@
963 \newcount\countRline
964   \countRline \z@
965

\@donereallinesL  We need to know the number of ‘real’ lines output (i.e., those that have been input
\@donetotallinesL by the user), and the total lines output (which includes any blank lines output for
\@donereallinesR synchronisation).
\@donetotallinesR 966 \newcount\@donereallinesL
967 \newcount\@donetotallinesL
968 \newcount\@donereallinesR
969 \newcount\@donetotallinesR
970

\do@lineL  The \do@lineL macro is called to do all the processing for a single line of left text.

971 \newcommand*{\do@lineL}{%
972   \letcs{\ifnumberpstart}{numberpstart@L\the\l@dpscl}%
973   \advance\countLline \@ne
974   \ifvbox\namebox{\l@dLcolrawbox\the\l@dpscl}%
975   {\vbadness=10000
976    \splittopskip=\z@
977    \do@lineLhook
978    \l@emptyd@ta
979    \global\setbox\one@line=\vsplit\namebox{\l@dLcolrawbox\the\l@dpscl}
980                                     to\baselineskip}%
981   \IfStrEq{\splitfirstmarks\parledgroup@}{begin}{\parledgroup@notes@startL}{\}
982   \unvbox\one@line \global\setbox\one@line=\lastbox
983   \getline@numL
984   \ifnum\@lock>\@ne%
985     \inserthangingsymboltrue%
986   \else%
987     \inserthangingsymbolfalse%
988   \fi
989   \setbox\l@dleftbox

```

```

990 \hb@xt@ \Lcolwidth{%
991   \affixline@num
992   \xifinlist{\the\l@dpscL}{\eled@sections@@}%
993   {\add@inserts\affixside@note}%
994   {\print@lineL}}%
995 \add@penaltiesL
996 \global\advance\@donereallinesL\@ne
997 \global\advance\@donetotallinesL\@ne
998 \else
999   \setbox\l@leftbox \hb@xt@ \Lcolwidth{\hspace*\Lcolwidth}}%
1000 \global\advance\@donetotallinesL\@ne
1001 \fi}
1002
1003

```

`\print@lineL` `\print@lineL` is for lines without a sectioning command. See `eledmac` definition of `\print@line` for handbook.

```

1004 \def\print@lineL{%
1005   \affixpstart@numL%
1006   \l@dld@ta %space kept for backward compatibility
1007   \add@inserts\affixside@note%
1008   \l@dlsn@te %space kept for backward compatibility
1009   {\ledllfill\hb@xt@ \wd\one@line{%
1010     \do@insidelineLhook%
1011     \ifluatex%
1012       \luatextextdir\l@luatextextdir@L%
1013     \fi%
1014     \new@lineL%
1015     \inserthangingsymbolL%
1016     \l@dunhbox@line{\one@line}}\correcthangingL\ledrlfill\l@drd@ta%
1017   \l@drrsn@te}}
1018

```

`\print@eledsectionL` `\print@eledsectionL` is for line with macro code.

```

1019 \def\print@eledsectionL{%%
1020   \addtocounter{pstartL}{-1}%
1021   \ifdefstring{\@eledsectnotoc}{L}{\ledsectnotoc}{%
1022     \ifdefstring{\@eledsectmark}{L}{\ledsectnomark}%
1023     \numdef{\temp@}{\l@dpscL-1}%
1024     \xifinlist{\temp@}{\eled@sections@@}{\@nobreaktrue}{\@nobreakfalse}%
1025     \@eled@sectioningtrue%
1026     \bgroup%
1027     \ifluatex%
1028       \luatextextdir\l@luatextextdir@L%
1029       \luatexpardir\l@luatexpardir@L%
1030       \luatexbodydir\l@luatexbodydir@L%
1031       \ifdefstring{\l@luatextextdir@L}{TRT}{\@RTLtrue}{%
1032     \fi%
1033     \csuse{eled@sectioning@\the\l@dpscL}%
1034     \egroup%

```

```

1035 \electedsectioningfalse%
1036 \global\csundef{eledsectioning@the\l@dpscl}%
1037 \if@RTL%
1038 \hspace{-3\paperwidth}%
1039 {\hbox{\l@dunhbox@line{\one@line}} \new@line}%
1040 \else%
1041 \hspace{3\paperwidth}%
1042 {\new@line \hbox{\l@dunhbox@line{\one@line}}}%
1043 \fi%
1044 \vskip\eledsection@correcting@skip%
1045 }
1046

```

`\dolineLhook` These high-level commands just redefine the low-level commands. They have to be used by user, without `\makeatletter`.

```

\doin sidelineLhook 1047 \newcommand*{\dolineLhook}[1]{\gdef\dolineLhook{#1}}%
\doin sidelineRhook 1048 \newcommand*{\dolineRhook}[1]{\gdef\dolineRhook{#1}}%
1049 \newcommand*{\doin sidelineLhook}[1]{\gdef\doin sidelineLhook{#1}}%
1050 \newcommand*{\doin sidelineRhook}[1]{\gdef\doin sidelineRhook{#1}}%
1051

```

`\do@lineLhook` Hooks, initially empty, into the respective `\do@line(L/R)` macros.

```

\dolineRhook 1052 \newcommand*{\do@lineLhook}{}
\doin sidelineLhook 1053 \newcommand*{\do@lineRhook}{}
\doin sidelineRhook 1054 \newcommand*{\do@insidelineLhook}{}
1055 \newcommand*{\do@insidelineRhook}{}
1056

```

`\do@lineR` The `\do@lineR` macro is called to do all the processing for a single line of right text.

```

1057 \newcommand*{\do@lineR}{%
1058 \letcs{\ifnumberpstart}{numberpstart@R\the\l@dpscl}%
1059 \ledRcol@true%
1060 \advance\countRline \@ne
1061 \ifvbox\namebox{\l@dRcolrawbox\the\l@dpscl}%
1062 {\vbadness=10000
1063 \splittopskip=\z@
1064 \do@lineRhook
1065 \l@emptyd@ta
1066 \global\setbox\one@lineR=\vsplit\namebox{\l@dRcolrawbox\the\l@dpscl}
1067 to\baselineskip}%
1068 \IfStrEq{\splitfirstmarks\parledgroup@}{begin}{\parledgroup@notes@startR}{-}
1069 \unvbox\one@lineR \global\setbox\one@lineR=\lastbox
1070 \getline@numR
1071 \ifnum\@lockR>\@ne%
1072 \inserthangingsymbolRtrue
1073 \else%
1074 \inserthangingsymbolRfalse%

```



```

1075 \fi%
1076 \setbox\l@drightbox
1077 \hb@xt@ \Rcolwidth{%
1078   \affixline@numR%
1079   \xifinlist{\the\l@dpscR}{\eled@sectionsR@@}%
1080   {\add@insertsR\affixside@noteR}%
1081   {\print@lineR}%
1082 }%
1083 \add@penaltiesR
1084 \global\advance\@donereallinesR\@ne
1085 \global\advance\@donetotallinesR\@ne
1086 \else
1087   \setbox\l@drightbox \hb@xt@ \Rcolwidth{\hspace*{\Rcolwidth}}
1088   \global\advance\@donetotallinesR\@ne
1089 \fi
1090 \ledRcol@false%
1091 }
1092
1093

```

```

\print@lineR
\print@eledsectionR

```

18.3 Line and page number computation

`\getline@numR` The `\getline@numR` macro determines the page and line numbers for the right text line we're about to send to the vertical list.

```

1094 \newcommand*\getline@numR{%
1095   \global\advance\absline@numR \@ne
1096   \do@actionsR
1097   \do@ballastR
1098   \ifledgroupnotesR\else\ifnumberline
1099     \ifsublines@
1100       \ifnum\sub@lockR<\tw@
1101         \global\advance\subline@numR \@ne
1102       \fi
1103     \else
1104       \ifnum\@lockR<\tw@
1105         \global\advance\line@numR \@ne
1106         \global\subline@numR \z@
1107       \fi
1108     \fi
1109 \fi
1110 \fi
1111 }
1112 \newcommand*\getline@numL{%
1113   \global\advance\absline@num \@ne
1114   \do@actions
1115   \do@ballast
1116   \ifledgroupnotesL\else\ifnumberline

```

```

1117 \ifsublines@
1118     \ifnum\sub@lock<\tw@
1119         \global\advance\subline@num \@ne
1120     \fi
1121 \else
1122     \ifnum\@lock<\tw@
1123         \global\advance\line@num \@ne
1124         \global\subline@num \z@
1125     \fi
1126 \fi
1127 \fi
1128 \fi
1129 }
1130
1131

```

`\do@ballastR` The real work in the line macros above is done in `\do@actions`, but before we plunge into that, let's get `\do@ballastR` out of the way.

```

1132 \newcommand*{\do@ballastR}{\global\ballast@count=\z@
1133 \begingroup
1134     \advance\absline@numR \@ne
1135     \ifnum\next@actionlineR=\absline@numR
1136         \ifnum\next@actionR>-1001
1137             \global\advance\ballast@count by -\c@ballast
1138         \fi
1139     \fi
1140 \endgroup}

```

`\do@actionsR` The `\do@actionsR` macro looks at the list of actions to take at particular right text absolute line numbers, and does everything that's specified for the current line.

`\do@actions@fixedcodeR`

`\do@actions@nextR`

It may call itself recursively and we use tail recursion, via `\do@actions@nextR` for this.

```

1141 \newcommand*{\do@actions@fixedcodeR}{%
1142     \ifcase\@l@dtmptcnta%
1143     \or% % 1001
1144         \global\sublines@true
1145     \or% % 1002
1146         \global\sublines@false
1147     \or% % 1003
1148         \global\@lockR=\@ne
1149     \or% % 1004
1150         \ifnum\@lockR=\tw@
1151             \global\@lockR=\thr@@
1152         \else
1153             \global\@lockR=\z@
1154         \fi
1155     \or% % 1005
1156         \global\sub@lockR=\@ne

```

```

1157 \or% % 1006
1158 \ifnum\sub@lockR=\tw@
1159 \global\sub@lockR=\thr@@
1160 \else
1161 \global\sub@lockR=\z@
1162 \fi
1163 \or% % 1007
1164 \l@dskipnumbertrue
1165 \else
1166 \led@warn@BadAction
1167 \fi}
1168
1169
1170 \newcommand*{\do@actionsR}{%
1171 \global\let\do@actions@nextR=\relax
1172 \@l@dtmpcntb=\absline@numR
1173 \ifnum\@l@dtmpcntb<\next@actionlineR\else
1174 \ifnum\next@actionR>-1001\relax
1175 \global\page@numR=\next@actionR
1176 \ifbypage@R
1177 \global\line@numR \z@ \global\subline@numR \z@
1178 \fi
1179 \else
1180 \ifnum\next@actionR<-4999\relax % 9/05 added relax here
1181 \@l@dtmpcnta=-\next@actionR
1182 \advance\@l@dtmpcnta by -5001\relax
1183 \ifsublines@
1184 \global\subline@numR=\@l@dtmpcnta
1185 \else
1186 \global\line@numR=\@l@dtmpcnta
1187 \fi
1188 \else
1189 \@l@dtmpcnta=-\next@actionR
1190 \advance\@l@dtmpcnta by -1000\relax
1191 \do@actions@fixedcodeR
1192 \fi
1193 \fi
1194 \ifx\actionlines@listR\empty
1195 \gdef\next@actionlineR{1000000}%
1196 \else
1197 \gl@p\actionlines@listR\to\next@actionlineR
1198 \gl@p\actions@listR\to\next@actionR
1199 \global\let\do@actions@nextR=\do@actionsR
1200 \fi
1201 \fi
1202 \do@actions@nextR}
1203

```

18.4 Line number printing

```

\l@dcalcnun \affixline@numR is the right text version of the \affixline@num macro.
\ch@cksub@l@ckR 1204
\ch@ck@l@ckR 1205 \providecommand*\l@dcalcnun}[3]{%
\fx@l@cksR 1206 \ifnum #1 > #2\relax
\affixline@numR 1207 \@l@tempcnta = #1\relax
1208 \advance\@l@tempcnta by -#2\relax
1209 \divide\@l@tempcnta by #3\relax
1210 \multiply\@l@tempcnta by #3\relax
1211 \advance\@l@tempcnta by #2\relax
1212 \else
1213 \@l@tempcnta=#2\relax
1214 \fi}
1215
1216 \newcommand*\ch@cksub@l@ckR}{%
1217 \ifcase\sub@lockR
1218 \or
1219 \ifnum\sublock@disp=\@ne
1220 \@l@tempcntb \z@ \@l@tempcnta \@ne
1221 \fi
1222 \or
1223 \ifnum\sublock@disp=\tw@
1224 \else
1225 \@l@tempcntb \z@ \@l@tempcnta \@ne
1226 \fi
1227 \or
1228 \ifnum\sublock@disp=\z@
1229 \@l@tempcntb \z@ \@l@tempcnta \@ne
1230 \fi
1231 \fi}
1232
1233 \newcommand*\ch@ck@l@ckR}{%
1234 \ifcase\@lockR
1235 \or
1236 \ifnum\lock@disp=\@ne
1237 \@l@tempcntb \z@ \@l@tempcnta \@ne
1238 \fi
1239 \or
1240 \ifnum\lock@disp=\tw@
1241 \else
1242 \@l@tempcntb \z@ \@l@tempcnta \@ne
1243 \fi
1244 \or
1245 \ifnum\lock@disp=\z@
1246 \@l@tempcntb \z@ \@l@tempcnta \@ne
1247 \fi
1248 \fi}
1249
1250 \newcommand*\fx@l@cksR}{%

```

```

1251 \ifcase\@lockR
1252 \or
1253   \global\@lockR \tw@
1254 \or \or
1255   \global\@lockR \z@
1256 \fi
1257 \ifcase\sub@lockR
1258 \or
1259   \global\sub@lockR \tw@
1260 \or \or
1261   \global\sub@lockR \z@
1262 \fi}
1263
1264
1265 \newcommand*{\affixline@numR}{%
1266 \ifledgroupnotesR\else\ifnumberline
1267 \ifl@dskipnumber
1268   \global\l@dskipnumberfalse
1269 \else
1270   \ifsublines@
1271     \@l@tempcntb=\subline@numR
1272     \l@dcalcnun{\subline@numR}{\c@firstsublinenumR}{\c@sublinenumincrementR}%
1273     \ch@cksub@lockR
1274   \else
1275     \@l@tempcntb=\line@numR
1276     \ifx\linenumberlist\empty
1277       \l@dcalcnun{\line@numR}{\c@firstlinenumR}{\c@linenumincrementR}%
1278     \else
1279       \@l@tempcnta=\line@numR
1280       \edef\rem@inder{\linenumberlist,\number\line@numR,}%
1281       \edef\sc@n@list{\def\noexpand\sc@n@list
1282         ###1,\number\@l@tempcnta,###2|{\def\noexpand\rem@inder{###2}}}%
1283       \sc@n@list\expandafter\sc@n@list\rem@inder|
1284       \ifx\rem@inder\empty\advance\@l@tempcnta\@ne\fi
1285     \fi
1286     \ch@ck@l@ckR
1287   \fi
1288   \ifnum\@l@tempcnta=\@l@tempcntb
1289     \if@twocolumn
1290       \if@firstcolumn
1291         \gdef\l@dld@ta{\llap{\leftlinenumR}}}%
1292       \else
1293         \gdef\l@drd@ta{\rlap{\rightlinenumR}}}%
1294       \fi
1295     \else
1296       \@l@tempcntb=\line@marginR
1297       \ifnum\@l@tempcntb>\@ne
1298         \advance\@l@tempcntb by\page@numR
1299       \fi
1300     \ifodd\@l@tempcntb

```

```

1301      \gdef\l@drd@ta{\rlap{{\rightlinenumR}}}%
1302      \else
1303      \gdef\l@dld@ta{\llap{{\leftlinenumR}}}%
1304      \fi
1305  \fi
1306  \fi
1307  \f@x@l@cksR
1308 \fi
1309 \fi
1310 \fi}

```

18.5 Pstart number printing in side

The printing of the pstart number is like in eldmac, with two differences :

- Some commands have versions suffixed by R or L.
- The `\affixpstart@num` and `\affixpstart@numR` commands are called in the `\Pages` command. Consequently, the `pstartL` and `pstartR` counters must be reset at the beginning of this command.

```

\affixpstart@numL
\affixpstart@numR 1311
  \leftpstartnumR 1312 \newcommand*{\affixpstart@numL}{%
\rightpstartnumR 1313 \ifsidepstartnum
  \leftpstartnumL 1314 \if@twocolumn
\rightpstartnumL 1315   \if@firstcolumn
  \ifpstartnumR 1316     \gdef\l@dld@ta{\llap{{\leftpstartnumL}}}%
1317     \else
1318     \gdef\l@drd@ta{\rlap{{\rightpstartnumL}}}%
1319     \fi
1320   \else
1321     \l@dtempcntb=\line@margin
1322     \ifnum\l@dtempcntb>\@ne
1323       \advance\l@dtempcntb \page@num
1324     \fi
1325     \ifodd\l@dtempcntb
1326     \gdef\l@drd@ta{\rlap{{\rightpstartnumL}}}%
1327     \else
1328     \gdef\l@dld@ta{\llap{{\leftpstartnumL}}}%
1329     \fi
1330   \fi
1331 \fi
1332 }
1333 \newcommand*{\affixpstart@numR}{%
1334 \ifsidepstartnum
1335 \if@twocolumn
1336   \if@firstcolumn
1337   \gdef\l@dld@ta{\llap{{\leftpstartnumR}}}%
1338   \else

```

```

1339     \gdef\l@drd@ta{\rlap{{\rightpstartnumR}}}%
1340     \fi
1341     \else
1342       \@l@tempcntb=\line@marginR
1343       \ifnum\@l@tempcntb>\@ne
1344         \advance\@l@tempcntb \page@numR
1345       \fi
1346       \ifodd\@l@tempcntb
1347         \gdef\l@drd@ta{\rlap{{\rightpstartnumR}}}%
1348       \else
1349         \gdef\l@dld@ta{\llap{{\leftpstartnumR}}}%
1350       \fi
1351     \fi
1352 \fi
1353 }
1354
1355 \newcommand*{\leftpstartnumL}{
1356 \ifpstartnum
1357 \thepstartL
1358 \kern\linenumsep\global\pstartnumfalse\fi
1359 }
1360 \newcommand*{\rightpstartnumL}{
1361 \ifpstartnum\kern\linenumsep
1362 \thepstartL
1363 \global\pstartnumfalse\fi
1364 }
1365 \newif\ifpstartnumR
1366 \pstartnumRtrue
1367 \newcommand*{\leftpstartnumR}{
1368 \ifpstartnumR
1369 \thepstartR
1370 \kern\linenumsep\global\pstartnumRfalse\fi
1371 }
1372 \newcommand*{\rightpstartnumR}{
1373 \ifpstartnumR\kern\linenumsep
1374 \thepstartR
1375 \global\pstartnumRfalse\fi
1376 }

```

18.6 Add insertions to the vertical list

`\inserts@listR` `\inserts@listR` is the list macro that contains the inserts that we save up for one right text paragraph.

```
1377 \list@create{\inserts@listR}
```

`\add@insertsR` The right text version.

```

\add@inserts@nextR 1378 \newcommand*{\add@insertsR}{%
1379   \global\let\add@inserts@nextR=\relax
1380   \ifx\inserts@listR\empty \else

```

```

1381 \ifx\next@insertR\empty
1382 \ifx\insertlines@listR\empty
1383 \global\noteschanged@true
1384 \gdef\next@insertR{100000}%
1385 \else
1386 \gl@p\insertlines@listR\to\next@insertR
1387 \fi
1388 \fi
1389 \ifnum\next@insertR=\absline@numR
1390 \gl@p\inserts@listR\to\@insertR
1391 \@insertR
1392 \global\let\@insertR=\undefined
1393 \global\let\next@insertR=\empty
1394 \global\let\add@inserts@nextR=\add@insertsR
1395 \fi
1396 \fi
1397 \add@inserts@nextR}
1398

```

18.7 Penalties

`\add@penaltiesL` `\add@penaltiesL` is the last macro used by `\do@lineL`. It adds up the club, widow, and interline penalties, and puts a single penalty of the appropriate size back into the paragraph; these penalties get removed by the `\vsplit` operation. `\displaywidowpenalty` and `\brokenpenalty` are not restored, since we have no easy way to find out where we should insert them.

In the code below, which is a virtual copy of the original `\add@penalties`, `\num@lines` is the number of lines in the whole paragraph, and `\par@line` is the line we're working on at the moment. The count `\@l@dttempcnta` is used to calculate and accumulate the penalty; it is initially set to the value of `\ballast@count`, which has been worked out in `\do@ballast`. Finally, the penalty is checked to see that it doesn't go below -10000 .

```

\newcommand*{\add@penaltiesR}{\@l@dttempcnta=\ballast@count
\ifnum\num@linesR>\@ne
\global\advance\par@lineR \@ne
\ifnum\par@lineR=\@ne
\advance\@l@dttempcnta by \clubpenalty
\fi
\@l@dttempcntb=\par@lineR \advance\@l@dttempcntb \@ne
\ifnum\@l@dttempcntb=\num@linesR
\advance\@l@dttempcnta by \widowpenalty
\fi
\ifnum\par@lineR<\num@linesR
\advance\@l@dttempcnta by \interlinepenalty
\fi
\fi
\ifnum\@l@dttempcnta=\z@
\relax

```



```

\else
  \ifnum\@l@dttempcnta>-10000
    \penalty\@l@dttempcnta
  \else
    \penalty -10000
  \fi
\fi}

```

This is for a single chunk. However, as we are probably dealing with several chunks at a time, the above is not really relevant. I think that it is likely with parallel text that there is no real need to add back any penalties; even if there was, they would have to match across the left and right lines. So, I end up with the following.

```

1399 \newcommand*{\add@penaltiesL}{ }
1400 \newcommand*{\add@penaltiesR}{ }
1401

```

18.8 Printing leftover notes

`\flush@notesR` The `\flush@notesR` macro is called after the entire right text has been sliced up and sent on to the vertical list.

```

1402 \newcommand*{\flush@notesR}{%
1403   \xloop
1404     \ifx\inserts@listR\empty \else
1405       \glp\inserts@listR\to\@insertR
1406       \@insertR
1407       \global\let\@insertR=\undefined
1408     \repeat}
1409

```

19 Footnotes

19.1 Normal footnote formatting

The `\printlines` macro prints the line numbers for a note—which, in the general case, is a rather complicated task. The seven parameters of the argument are the line numbers as stored in `\l@d@nums`, in the form described on page 65 of *eledmac*’ handbook: the starting page, line, and sub-line numbers, followed by the ending page, line, and sub-line numbers, and then the font specifier for the lemma.

`\printlinesR` This is the right text version of `\printlines` and takes account of `\Rlineflag`.
`\ledsavedprintlines` Just in case, `\ledsavedprintlines` is a copy of the original `\printlines`.
 Just a reminder of the arguments:

<code>\printlinesR</code>	<code>#1</code>	<code> </code>	<code>#2</code>	<code> </code>	<code>#3</code>	<code> </code>	<code>#4</code>	<code> </code>	<code>#5</code>	<code> </code>	<code>#6</code>	<code> </code>	<code>#7</code>
<code>\printlinesR</code>	start-page		line		subline		end-page		line		subline		font

```

1410 \def\printlinesR#1|#2|#3|#4|#5|#6|#7|{\begingroup
1411   \setprintlines{#1}{#2}{#3}{#4}{#5}{#6}%

```

```

1412 \ifl@d@pnum #1\fullstop\fi
1413 \ifledplinenum \linenumr@p{#2}\Rlineflag\else \sympmlinenum\fi
1414 \ifl@d@ssub \fullstop \sublinenumr@p{#3}\fi
1415 \ifl@d@dash \endashchar\fi
1416 \ifl@d@pnum #4\fullstop\fi
1417 \ifl@d@elin \linenumr@p{#5}\Rlineflag\fi
1418 \ifl@d@esl \ifl@d@elin \fullstop\fi \sublinenumr@p{#6}\fi
1419 \endgroup}
1420
1421 \let\ledsavedprintlines\printlines
1422

```

19.2 Footnotes output specific to \Pages

`\print@Xnotes@forpages` The `\onlyXside` and `\onlysideX` hooks for `\Pages` allow notes to be printed either in left or right pages only. The implementation of such features is delegated to `\print@Xnotes@forpages`, which replaces `\print@Xnotes` inside `\Pages`. Here is how we proceed²:

- If notes are to be printed in both sides, we just proceed the usual way: print the foot starts for the series, then the foot group.
- If notes are to be printed in the left side, we do these prints only for even pages ; if notes are to be printed in the right side, we do these prints only for odd pages.
- However, that is not enough. Because the problem does not only consists in printing notes in any particular page. It is also not to put aside room for notes in the pages where we don't want to print them. To take an example: if some note in the left side is too long by 160pt to be printed in full in the left page, we do not want to put aside 160pt a space for it in the following right page.
- To solve this problem, we change the magnification factor associated with notes before going to the next page. If we start a page where no notes are supposed to be printed, the magnification counter is set to 0. We also set the note skip to 0pt. Before starting a new page where these notes are supposed to be printed, we reset these counter and skip to their default values. (About these counter and skip, read *TeXbook* p. 122-125).
- There still remains a last problem. This problem is quite complex to understand, so an example will speak for itself. Suppose we allow 10 lines of notes by page. Suppose a long note, be it 25 lines, which needs three pages to be printed. Suppose it must be printed only on left pages, namely odd pages.

On p. 2, the first 10 lines of the notes are printed. On p. 3, the box associated to the notes contains 10 lines. However, as we are in a right page, we don't void this box. So \TeX will keep its content for the pages to come. However,

²See <http://tex.stackexchange.com/a/230332/7712>.

on p. 4 it will also add one line in the footnote box, because in any case, `TeX` adds some content in the box when preparing the output routines, even if there is some content left in this box from the previous pages. So the lines in the note box at p. 4 will be $10 + 1 = 11$. There is one line which should not be there. Furthermore, as the box size is for 10 lines and not for 11 lines, this last line will be glued to the previous one.

To fix this double issue:

- For the pages where notes must be NOT printed, we allow to every note box one line less than it ought to be. In our example, that means that we allow `TeX` to add only $10 - 1 = 9$ line in the note box on p. 3. Before shifting to the pages where notes must be printed, we allow to every notes the expected number of lines. In our example, that means that we allow `TeX` to add 10 lines in the note box on p. 4. As on p. 3 only 9 lines were allowed, that means note box of p. 4 will contain $9 + 1 = 10$ lines. So the “one line too many” problem is solved.
- Still remains the “glue” problem. We solve it by recreating a clean note box. We split the one which is created by `TeX` to get the next line printed. Then, we create the new box, by bringing together the first part and the last part of the splitted box, adding some skip between them. That is achieved by `\correct@Xfootins@box` (or `\correct@footinsX@box` for familiar notes).

The code to print critical notes, when processing `\Pages`

```
1423 \newcommand\print@Xnotes@forpages[1]{%
```

First case: notes are for both sides. Just print the note start and the note group

```
1424   \ifcseempty{onlyXside@#1}{}%
1425       \csuse{#1footstart}{#1}%
1426       \csuse{#1footgroup}{#1}%
1427   }%
```

Second case: notes are for one side only. First test if we are in a page where they must be printed.

```
1428   {%
1429       \ifboolexpr{%
1430           ((test {\ifcsstring{onlyXside@#1}{L}} and not test{\ifnumodd{\c@page}}))%
1431           or%
1432           (test {\ifcsstring{onlyXside@#1}{R}} and test{\ifnumodd{\c@page}}))%
1433       }%
```

If we are in a page where notes must be printed, print the notes, after having made the corrections which are needed for boxes.

```
1434   {%
1435       \correct@Xfootins@box{#1}%
1436       \csuse{#1footstart}{#1}%
1437       \csuse{#1footgroup}{#1}%
```

Then, say not to keep room for notes in the next page.

```
1438      \global\count\csuse{#1footins}=0%
1439      \global\skip\csuse{#1footins}=0pt%
```

And also, allow one line less for notes in the next page.

```
1440      \csuse{Xnotefontsize@#1}%
1441      \global\advance\dimen\csuse{#1footins} by -\baselineskip%
```

Now we have printed the notes. So we put aside this fact.

```
1442      \global\boolfalse{keepforXside@#1}%
1443      }%
```

In case we are on a page where notes must NOT be printed. First, memorize that we have not printed the notes, despite having some to print.

```
1444      {%
1445      \global\booltrue{keepforXside@#1}%
```

Then restore expected rooms for notes on the next page.

```
1446      \global\count\csuse{#1footins}=\csuse{default@#1footins}%
1447      \global\skip\csuse{#1footins}=\csuse{beforeXnotes@#1}%
```

Last but not least, restore the normal line number allowed to notes for the following page.

```
1448      \bgroup%
1449      \csuse{Xnotefontsize@#1}%
1450      \global\advance\dimen\csuse{#1footins} by \baselineskip%
1451      \egroup%
1452 % End of \cs{print@Xnotes@forpages}.
1453      }%
1454      }%
1455      }%
```

Now, `\correct@Xfootins@box`, to fix problem of last line being glued to the previous one.

```
1456 \newcommand{\correct@Xfootins@box}[1]{%
```

We need to make correction only in case we have not printed any note in the previous page, although there was to be “normally” printed.

```
1457 \ifbool{keepforXside@#1}{%
```

Some setting needed to do the right splitting.

```
1458      \csuse{Xnotefontsize@#1}%
1459      \splittopskip=0pt%
```

And now, split the last line, and push in the right place.

```
1460      \global\setbox\csuse{#1footins}=\vbox{%
1461      \vsplit\csuse{#1footins} to \dimexpr\ht\csuse{#1footins}-1pt\relax%
1462      \vskip \dimexpr-0.5\baselineskip-0.5\lineskip-0.5pt\relax%
1463      \unvbox\csuse{#1footins}%
1464      }%
```

End of the macro.

```

1465 }{}%
1466 }%

And now, the same for familiar footnotes.
1467 \newcommand\print@notesX@forpages[1]{%
1468   \ifcseempty{onlysideX@#1}{%
1469     \csuse{footstart#1}{#1}%
1470     \csuse{footgroup#1}{#1}%
1471   }%
1472   {%
1473     \ifboolexpr{%
1474       ((test {\ifcsstring{onlysideX@#1}{L}} and not test{\ifnumodd{\c@page}}))%
1475       or%
1476       (test {\ifcsstring{onlysideX@#1}{R}} and test{\ifnumodd{\c@page}}))%
1477     }%
1478     {%
1479       \correct@footinsX@box{#1}%
1480       \csuse{footstart#1}{#1}%
1481       \csuse{footgroup#1}{#1}%
1482       \global\count\csuse{footins#1}=0%
1483       \global\skip\csuse{footins#1}=0pt%
1484       \csuse{notefontsizeX@#1}%
1485       \global\advance\dimen\csuse{footins#1} by -\baselineskip%
1486       \global\boolfalse{keepforsideX@#1}%
1487     }%
1488     {%
1489       \global\booltrue{keepforsideX@#1}%
1490       \global\count\csuse{footins#1}=\csuse{default@footins#1}%
1491       \global\skip\csuse{footins#1}=\csuse{beforenotesX@#1}%
1492       \bgroup%
1493         \csuse{notefontsizeX@#1}%
1494         \global\advance\dimen\csuse{footins#1} by \baselineskip%
1495       \egroup%
1496     }%
1497   }%
1498 }%
1499 \newcommand{\correct@footinsX@box}[1]{%
1500   \ifbool{keepforsideX@#1}{%
1501     \csuse{notefontsizeX@#1}%
1502     \splittopskip=0pt%
1503     \global\setbox\csuse{footins#1}=\vbox{%
1504       \vsplit\csuse{footins#1} to \dimexpr\ht\csuse{footins#1}-1pt\relax%
1505       \vskip \dimexpr-0.5\baselineskip-0.5\lineskip-0.5pt\relax%
1506       \unvbox\csuse{footins#1}%
1507     }%
1508   }{}%
1509 }%

```

20 Cross referencing

`\labelref@listR` Set up a new list, `\labelref@listR`, to hold the page, line and sub-line numbers for each label in right text.

```
1510 \list@create{\labelref@listR}
1511
```

`\edlabel` Since version 1.18.0, this command is defined only one time in `eledmac`, including features for `eledpar`.

`\l@dmake@labelsR` This is the right text version of `\l@dmake@labels`, taking account of `\Rlineflag`.

```
1512 \def\l@dmake@labelsR#1|#2|#3|#4|#5{%
1513   \expandafter\ifx\csname the@label#5\endcsname \relax\else
1514     \led@warn@DuplicateLabel{#4}%
1515   \fi
1516   \expandafter\gdef\csname the@label#5\endcsname{#1|#2\Rlineflag|#3|#4}%
1517   \ignorespaces}
1518 \AtBeginDocument{%
1519   \def\l@dmake@labelsR#1|#2|#3|#4|#5{%}
1520 }
1521
```

`\@lab` The `\@lab` command, which appears in the `\linenum@out` file, appends the current values of page, line and sub-line to the `\labelref@list`. These values are defined by the earlier `\@page`, `\@nl`, and the `\sub@on` and `\sub@off` commands appearing in the `\linenum@out` file.

```
1522 \renewcommand*{\@lab}{%
1523   \ifledRcol
1524     \xright@appenditem{\linenumr@p{\line@numR}}{|%
1525     \ifsublines@ \sublinenumr@p{\subline@numR}\else 0\fi}%
1526     \to\labelref@listR
1527   \else
1528     \xright@appenditem{\linenumr@p{\line@num}}{|%
1529     \ifsublines@ \sublinenumr@p{\subline@num}\else 0\fi}%
1530     \to\labelref@list
1531   \fi}
1532
```

21 Side notes

Regular `\marginpars` do not work inside numbered text — they don't produce any note but do put an extra unnumbered blank line into the text.

`\sidenote@marginR` Specifies which margin sidenotes can be in.

```
\sidenotemargin* 1533 \WithSuffix\newcommand\sidenotemargin*[1]{%
1534   \l@dgetsidenote@margin{#1}
1535   \global\sidenote@marginR=\@l@tempcntb
1536   \global\sidenote@margin=\@l@tempcntb
```

```

1537 }
1538 \newcount\sidenote@marginR
1539 \global\sidenote@margin=\@ne
1540

```

`\affixside@noteR` The right text version of `\affixside@note`.

```

1541 \newcommand*{\affixside@noteR}{%
1542   \def\sidenotecontent@{%
1543     \numdef{\itemcount@}{0}%
1544     \def\do##1{%
1545       \ifnumequal{\itemcount@}{0}%
1546         {%
1547           \appto\sidenotecontent@{##1}}% Not print not separator before the 1st note
1548           {\appto\sidenotecontent@{\sidenotesep ##1}%
1549           }%
1550         \numdef{\itemcount@}{\itemcount@+1}%
1551       }%
1552       \dolistloop{\l@dcstotetext}%
1553       \ifnumgreater{\itemcount@}{1}{\led@err@ManySidenotes}{}%
1554       \gdef\@templ@d{%
1555         \gdef\@templ@n{\l@dcstotetext\l@dcstotetext@l\l@dcstotetext@r}%
1556         \ifx\@templ@d\@templ@n \else%
1557           \if@twocolumn%
1558             \if@firstcolumn%
1559               \setl@dlp@rbox{##1}{\sidenotecontent@}%
1560             \else%
1561               \setl@drp@rbox{\sidenotecontent@}%
1562             \fi%
1563           \else%
1564             \l@dttempcntb=\sidenote@marginR%
1565             \ifnum\l@dttempcntb>\@ne%
1566               \advance\l@dttempcntb by\page@numR%
1567             \fi%
1568             \ifodd\l@dttempcntb%
1569               \setl@drp@rbox{\sidenotecontent@}%
1570               \gdef\sidenotecontent@{%
1571                 \numdef{\itemcount@}{0}%
1572                 \dolistloop{\l@dcstotetext@l}%
1573                 \ifnumgreater{\itemcount@}{1}{\led@err@ManyLeftnotes}{}%
1574                 \setl@dlp@rbox{\sidenotecontent@}%
1575               \else%
1576                 \setl@dlp@rbox{\sidenotecontent@}%
1577                 \gdef\sidenotecontent@{%
1578                   \numdef{\itemcount@}{0}%
1579                   \dolistloop{\l@dcstotetext@r}%
1580                   \ifnumgreater{\itemcount@}{1}{\led@err@ManyRightnotes}{}%
1581                   \setl@drp@rbox{\sidenotecontent@}%
1582                 \fi%
1583               \fi%

```

```

1584 \fi%
1585 }
1586

```

22 Familiar footnotes

`\l@dbfnote` `\l@dbfnote` adds the footnote to the insert list, and `\vl@dbfnote` calls the original `\@footnotetext`.

```

1587 \renewcommand{\l@dbfnote}[1]{%
1588   \ifnumberedpar@
1589   \gdef\@tag{#1\relax}%
1590   \ifledRcol%
1591     \xright@appenditem{\noexpand\vl@dbfnote{\@expandonce\@tag}}{\@thefnmark}}%
1592     \to\inserts@listR
1593     \global\advance\insert@countR \@ne%
1594   \else%
1595     \xright@appenditem{\noexpand\vl@dbfnote{\@expandonce\@tag}}{\@thefnmark}}%
1596     \to\inserts@list
1597     \global\advance\insert@count \@ne%
1598   \fi
1599 \fi\ignorespaces}
1600

```

`\normalbfnoteX`

```

1601 \renewcommand{\normalbfnoteX}[2]{%
1602   \ifnumberedpar@
1603   \ifledRcol%
1604     \ifluatex
1605       \footnotelang@lua[R]%
1606     \fi
1607     \@ifundefined{xpg@main@language}%if polyglossia
1608     {}%
1609     {\footnotelang@poly[R]}%
1610     \protected@xdef\thisfootnote{\csuse{thefootnote#1}}%
1611     \xright@appenditem{\noexpand\vbfnoteX{#1}{#2}{\@expandonce\thisfootnote}}%
1612     \to\inserts@listR
1613     \global\advance\insert@countR \@ne%
1614   \else%
1615     \ifluatex
1616       \footnotelang@lua%
1617     \fi
1618     \@ifundefined{xpg@main@language}%if polyglossia
1619     {}%
1620     {\footnotelang@poly}%
1621     \protected@xdef\thisfootnote{\csuse{thefootnote#1}}%
1622     \xright@appenditem{\noexpand\vbfnoteX{#1}{#2}{\@expandonce\thisfootnote}}%
1623     \to\inserts@list
1624     \global\advance\insert@count \@ne%

```



```

1625     \fi
1626     \fi\ignorespaces}
1627

```

23 Verse

Like in eledmac, the insertion of hangingsymbol is base on \ifinserthangingsymbol, and, for the right side, on \ifinserthangingsymbolR.

```

\inserthangingsymbolL
\inserthangingsymbolR 1628 \newif\ifinserthangingsymbolR
1629 \newcommand{\inserthangingsymbolL}{%
1630 \ifinserthangingsymbol%
1631     \ifinstanzaL%
1632         \hangingsymbol%
1633     \fi%
1634 \fi}
1635 \newcommand{\inserthangingsymbolR}{%
1636 \ifinserthangingsymbolR%
1637     \ifinstanzaR%
1638         \hangingsymbol%
1639     \fi%
1640 \fi}

```

When a verse is hanged, the column separator is shifted. To prevent it, the \do@lineL and \do@lineR commands call \correcthangingL and \correcthangingR commands. These commands insert horizontal skip which length is equal to the hang indent.

```

\correcthangingL
\correcthangingR 1641 \newcommand{\correcthangingL}{%
1642 \ifl@dpaging\else%
1643     \ifinstanzaL%
1644         \ifinserthangingsymbol%
1645             \hskip \@ifundefined{sza@0@}{0}{\expandafter%
1646                 \noexpand\csname sza@0@\endcsname}\stanzaindentbase%
1647         \fi%
1648     \fi%
1649 \fi}
1650
1651 \newcommand{\correcthangingR}{%
1652 \ifl@dpaging\else%
1653     \ifinstanzaR%
1654         \ifinserthangingsymbolR%
1655             \hskip \@ifundefined{sza@0@}{0}{\expandafter%
1656                 \noexpand\csname sza@0@\endcsname}\stanzaindentbase%
1657         \fi%
1658     \fi%
1659 \fi}

```

Before we can define the main stanza macros we need to be able to save and reset the category code for `&`. To save the current value we use `\next` from the `\loop` macro.

```
1660 \chardef\next=\catcode'\&
1661 \catcode'\&=\active
1662
```

astanza This is roughly an environmental form of `\stanza`, which treats its stanza-like contents as a single chunk.

```
1663 \newenvironment{astanza}{%
1664   \startstanzahook
1665   \catcode'\&\active
1666   \global\stanza@count\@ne\stanza@modulo\@ne
1667   \ifnum\usernamecount{sza@0@}=\z@
1668     \let\stanza@hang\relax
1669     \let\endlock\relax
1670   \else
1671     \rightskip\z@ plus 1fil\relax
1672   \fi
1673   \ifnum\usernamecount{szp@0@}=\z@
1674     \let\sza@penalty\relax
1675   \fi
1676   \def&{%
1677     \endlock\mbox{}}%
1678     \sza@penalty
1679     \global\advance\stanza@count\@ne
1680     \@astanza@line}%
1681   \def\&{\@stopastanza}%
1682   \pstart
1683   \@astanza@line
1684 }{}
1685
```

\@stopastanza This command is called by `\&` in `astanza` environment. It allows optional arguments.

```
1686 \newcommandx{\@stopastanza}[1][1,usedefault]{%
1687   \endlock\mbox{}}%
1688   \pend[#1]%
1689   \endstanzaextra%
1690 }%
```

\@astanza@line This gets put at the start of each line in the environment. It sets up the paragraph style — each line is treated as a paragraph.

```
1691 \newcommand*{\@astanza@line}{%
1692   \ifnum\value{stanzaindentrepetition}=0
1693     \parindent=\csname sza@\number\stanza@count
1694       @\endcsname\stanzaindentbase
1695   \else
1696     \parindent=\csname sza@\number\stanza@modulo
```

```

1697             @\endcsname\stanzaindentbase
1698         \managestanza@modulo
1699     \fi
1700 \par
1701 \stanza@hang%\mbox{}%
1702 \ignorespaces}
1703

```

Lastly reset the modified category codes.

```

1704 \catcode'\&=\next
1705

```

24 Naming macros

The L^AT_EX kernel provides `\@namedef` and `\@namuse` for defining and using macros that may have non-letters in their names. We need something similar here as we are going to need and use some numbered boxes and counters.

`\newnamebox` A set of macros for creating and using ‘named’ boxes; the macros are called after the regular box macros, but including the string ‘name’.

```

\unhnamebox 1706 \providecommand*\newnamebox}[1]{%
\unvnamebox 1707 \expandafter\newbox\csname #1\endcsname}
\namebox 1708 \providecommand*\setnamebox}[1]{%
1709 \expandafter\setbox\csname #1\endcsname}
1710 \providecommand*\unhnamebox}[1]{%
1711 \expandafter\unhbox\csname #1\endcsname}
1712 \providecommand*\unvnamebox}[1]{%
1713 \expandafter\unvbox\csname #1\endcsname}
1714 \providecommand*\namebox}[1]{%
1715 \csname #1\endcsname}
1716

```

`\newnamecount` Macros for creating and using ‘named’ counts.

```

\usenamecount 1717 \providecommand*\newnamecount}[1]{%
1718 \expandafter\newcount\csname #1\endcsname}
1719 \providecommand*\usenamecount}[1]{%
1720 \csname #1\endcsname}
1721

```

25 Counts and boxes for parallel texts

In sequential text, each chunk (that enclosed by `\pstart ... \pend`) is put into a box called `\raw@text` and then immediately printed, resulting in the box being emptied and ready for the next chunk. For parallel processing multiple boxes are needed as printing is delayed. We also need extra counters for various things.

`\maxchunks` The maximum number of chunk pairs before printing has to be called for. The
`\l@dc@maxchunks` default is 5120 chunk pairs.

```

1722 \newcount\l@dc@maxchunks
1723 \newcommand{\maxchunks}[1]{\l@dc@maxchunks=#1}
1724   \maxchunks{5120}
1725

```

`\l@dnumpstartsL` The numbers of left and right chunks. `\l@dnumpstartsL` is defined in `eledmac`.
`\l@dnumpstartsR`

```

1726 \newcount\l@dnumpstartsR
1727

```

`\l@pscl` A couple of scratch counts for use in left and right texts, respectively.
`\l@pscr`

```

1728 \newcount\l@dpscL
1729 \newcount\l@dpscR
1730

```

`\l@dsetuprawboxes` This macro creates `\maxchunks` pairs of boxes for left and right chunks. The boxes
are called `\l@dLcolrawbox1`, `\l@dLcolrawbox2`, etc.

```

1731 \newcommand*{\l@dsetuprawboxes}{%
1732   \@l@tempcntb=\l@dc@maxchunks
1733   \loop\ifnum\@l@tempcntb>\z@
1734     \newnamebox{\l@dLcolrawbox\the\@l@tempcntb}
1735     \newnamebox{\l@dRcolrawbox\the\@l@tempcntb}
1736     \advance\@l@tempcntb \m@ne
1737   \repeat}
1738

```

`\l@dsetupmaxlinecounts` To be able to synchronise left and right texts we need to know the maximum num-
`\l@dzeromaxlinecounts` ber of text lines there are in each pair of chunks. `\l@dsetupmaxlinecounts` creates
`\maxchunks` new counts called `\l@dmaxlinesinpar1`, etc., and `\l@dzeromaxlinecounts`
zeroes all of them.

```

1739 \newcommand*{\l@dsetupmaxlinecounts}{%
1740   \@l@tempcntb=\l@dc@maxchunks
1741   \loop\ifnum\@l@tempcntb>\z@
1742     \newnamecount{\l@dmaxlinesinpar\the\@l@tempcntb}
1743     \advance\@l@tempcntb \m@ne
1744   \repeat}
1745 \newcommand*{\l@dzeromaxlinecounts}{%
1746   \begingroup
1747   \@l@tempcntb=\l@dc@maxchunks
1748   \loop\ifnum\@l@tempcntb>\z@
1749     \global\usenamecount{\l@dmaxlinesinpar\the\@l@tempcntb}=\z@
1750     \advance\@l@tempcntb \m@ne
1751   \repeat
1752   \endgroup}
1753

```

Make sure that all these are set up. This has to be done after the user has had an opportunity to change `\maxchunks`.

```

1754 \AtBeginDocument{%
1755   \l@dssetuprawboxes
1756   \l@dssetupmaxlinecounts
1757   \l@dzzeromaxlinecounts
1758   \l@dnumpstartsL=\z@
1759   \l@dnumpstartsR=\z@
1760   \l@dpscL=\z@
1761   \l@dpscR=\z@}
1762

```

26 Fixing babel

With parallel texts there is the possibility that the two sides might use different languages via `babel`. On the other hand, `babel` might not be called at all (even though it might be already built into the format).

With the normal sequential text each line is initially typeset in the current language environment, and then it is output at which time its attachments are typeset (in the same language environment. In the parallel case lines are typeset in their current language but an attachment might be typeset outside the language environment of its line if the left and right side languages are different. To counter this, we have to make sure that the correct language is used at the proper times.

```

\ifl@dusedbabel A flag for checking if babel has been used as a package.
\l@dusedbabelfalse 1763 \newif\ifl@dusedbabel
\l@dusedbabeltrue
\ifl@dsamelang Suppress \ifl@dsamelang which didn't work and was not logical, because both
columns could have the same language but not the main language of the document.

\l@dchecklang

\l@dbbl@set@language In babel the macro \bbl@set@language{<lang>} does the work when the language
<lang> is changed via \selectlanguage. Unfortunately for me, if it is given an
argument in the form of a control sequence it strips off the \ character rather than
expanding the command. I need a version that accepts an argument in the form
\lang without it stripping the \.
1764 \newcommand*{\l@dbbl@set@language}[1]{%
1765   \edef\language{#1}%
1766   \select@language{\language}%
1767   \if@files
1768     \protected@write\@auxout{}\string\select@language{\language}%
1769     \addtocontents{toc}{\string\select@language{\language}%
1770     \addtocontents{lof}{\string\select@language{\language}%
1771     \addtocontents{lot}{\string\select@language{\language}%
1772   \fi}
1773

```

The rest of the setup has to be postponed until the end of the preamble when we know if `babel` has been used or not. However, for now assume that it has not been used.

`\selectlanguage` `\selectlanguage` is a babel command. `\theledlanguageL` and `\theledlanguageR` are the names of the languages of the left and right texts. `\l@duselanguage` is similar to `\selectlanguage`.

```
\theledlanguageR 1774 \providecommand{\selectlanguage}[1]{%
1775 \newcommand*{\l@duselanguage}[1]{%
1776 \gdef\theledlanguageL{}
1777 \gdef\theledlanguageR{}
1778
```

Now do the babel fix or polyglossia, if necessary.

```
1779 \AtBeginDocument{%
1780 \ifundefined{xpg@main@language}{%
1781 \ifundefined{bbl@main@language}{%
```

Either babel has not been used or it has been used with no specified language.

```
1782 \l@dusedbabelfalse
1783 \renewcommand*{\selectlanguage}[1]{}%
```

Here we deal with the case where babel has been used. `\selectlanguage` has to be redefined to use our version of `\bbl@set@language` and to store the left or right language.

```
1784 \l@dusedbabeltrue
1785 \let\l@doldselectlanguage\selectlanguage
1786 \let\l@doldbbl@set@language\bbl@set@language
1787 \let\bbl@set@language\l@dbbl@set@language
1788 \renewcommand{\selectlanguage}[1]{%
1789 \l@doldselectlanguage{#1}%
1790 \ifl@Rcol \gdef\theledlanguageR{#1}%
1791 \else \gdef\theledlanguageL{#1}%
1792 \fi}
```

`\l@duselanguage` simply calls the original `\selectlanguage` so that `\theledlanguageL` and `\theledlanguageR` are unaltered.

```
1793 \renewcommand*{\l@duselanguage}[1]{%
1794 \l@doldselectlanguage{#1}}
```

Lastly, initialise the left and right languages to the current babel one.

```
1795 \gdef\theledlanguageL{\bbl@main@language}%
1796 \gdef\theledlanguageR{\bbl@main@language}%
1797 }%
1798 }
```

If on Polyglossia

```
1799 { \let\old@otherlanguage\otherlanguage%
1800 \renewcommand{\otherlanguage}[2][1]{%
1801 \selectlanguage{#1}{#2}%
1802 \ifl@Rcol \gdef\theledlanguageR{#2}%
1803 \else \gdef\theledlanguageL{#2}%
1804 \fi}%
1805 \let\l@duselanguage\select@language%
1806 \gdef\theledlanguageL{\xpg@main@language}%
1807 \gdef\theledlanguageR{\xpg@main@language}%
```

That's it.

1808 }}

`\if@pstarts` `\check@pstarts` returns `\@pstartstrue` if there are any unprocessed chunks.

```
\@pstartstrue 1809 \newif\if@pstarts
\@pstartsfalse 1810 \newcommand*{\check@pstarts}{%
\check@pstarts 1811 \@pstartsfalse
1812 \ifnum\l@dnumpstartsL>\l@dpscL
1813 \@pstartstrue
1814 \else
1815 \ifnum\l@dnumpstartsR>\l@dpscR
1816 \@pstartstrue
1817 \fi
1818 \fi
1819 }
1820
```

`\ifaraw@text` `\checkraw@text` checks whether the current Left or Right box is void or not. If `\araw@texttrue` one or other is not void it sets `\araw@texttrue`, otherwise both are void and it `\araw@textfalse` sets `\araw@textfalse`.

```
\checkraw@text 1821 \newif\ifaraw@text
1822 \newcommand*{\checkraw@text}{%
1823 \araw@textfalse
1824 \ifvbox\namebox{1@dLcolrawbox\the\l@dpscL}
1825 \araw@texttrue
1826 \else
1827 \ifvbox\namebox{1@dRcolrawbox\the\l@dpscR}
1828 \araw@texttrue
1829 \fi
1830 \fi
1831 }
1832
```

`\@writelinesinparL` These write the number of text lines in a chunk to the section files, and then `\@writelinesinparR` afterwards zero the counter.

```
1833 \newcommand*{\@writelinesinparL}{%
1834 \edef\next{%
1835 \write\linenum@out{\string\@pend[\the\@donereallinesL]}}%
1836 \next
1837 \global\@donereallinesL \z@}
1838 \newcommand*{\@writelinesinparR}{%
1839 \edef\next{%
1840 \write\linenum@outR{\string\@pendR[\the\@donereallinesR]}}%
1841 \next
1842 \global\@donereallinesR \z@}
1843
```

27 Parallel columns

`\@eledsectionL` The `\@eledsectionL` and `\@eledsectionR` will keep the sections' title.

`\@eledsectionR` 1844 `\newsavebox{\@eledsectionL}%`

1845 `\newsavebox{\@eledsectionR}%`

`\Columns` The `\Columns` command results in the previous Left and Right texts being typeset in matching columns. There should be equal numbers of chunks in the left and right texts.

```
1846 \newcommand*{\Columns}{%
1847   \l@dpentingcolumnstrue%
1848   \eledsection@correcting@skip=-\baselineskip% Correction for sections' titles
1849   \ifnum\l@dnumstartL=\l@dnumstartR\else
1850     \led@err@BadLeftRightPstarts{\the\l@dnumstartL}{\the\l@dnumstartR}%
1851   \fi
```

Start a group and zero counters, etc.

```
1852 \begingroup
1853   \l@dzeropenalties
1854   \endgraf\global\l@num@lines=\prevgraf
1855   \global\l@num@linesR=\prevgraf
1856   \global\l@par@line=\z@
1857   \global\l@par@lineR=\z@
1858   \global\l@dpscL=\z@
1859   \global\l@dpscR=\z@
```

Check if there are chunks to be processed, and process them two by two (left and right pairs).

```
1860   \check@pstarts
1861   \loop\if@pstarts
1862     \global\l@pstartnumtrue
1863     \global\l@pstartnumRtrue
```

Increment `\l@dpscL` and `\l@dpscR` which here count the numbers of left and right chunks. Also restore the value of the public `pstart` counters.

```
1864   \global\advance\l@dpscL \@ne
1865   \global\advance\l@dpscR \@ne
1866   \restore@pstartL@pc%
1867   \restore@pstartR@pc%
```

Check if there is text yet to be processed in at least one of the two current chunks, and also whether the left and right languages are the same

```
1868   \checkraw@text
1869 {   \loop\ifaraw@text
```

Grab the next pair of left and right text lines and output them, swapping languages if they differ, adding section title if needed.

```
1870     \l@duselanguage{\theledlanguageL}%
1871     \do@lineL
1872     \xifinlist{\the\l@dpscL}{\eled@sections@}
```



```

1873      {%
1874      \ifdefstring{\@eledsectmark}{L}%
1875      {\csuse{eled@sectmark@the\l@dpscl}%
1876      }{%}%
1877      \global\csundef{eled@sectmark@the\l@dpscl}%
1878      \savebox{\@eledsectionL}{\parbox[t][t]{\Lcolwidth}{\vbox{\print@eledsectionL}}}%\vb
1879      }%
1880      }{%}%
1881      \l@duselanguage{\theledlanguageR}%
1882      \do@lineR
1883      \xifinlist{\the\l@dpscr}{\eled@sectionsR@@}
1884      {%
1885      \ifdefstring{\@eledsectmark}{R}%
1886      {\csuse{eled@sectmark@the\l@dpscr R}%
1887      }{%}%
1888      \global\csundef{eled@sectmark@the\l@dpscr R}%
1889      \savebox{\@eledsectionR}{\parbox[t][t]{\Rcolwidth}{\vbox{\print@eledsectionR}}}%\vb
1890      }{%}%
1891      \hb@xt@ \hsize{%
1892      \ifdefstring{\columns@position}{L}{\hfill }%
1893      \unhbox\l@dleftbox%
1894      \ifhbox\@eledsectionL%
1895      \usebox{\@eledsectionL}%
1896      \fi%
1897      \print@columnseparator%
1898      \unhbox\l@drightbox%
1899      \ifhbox\@eledsectionR%
1900      \usebox{\@eledsectionR}%
1901      \fi%
1902      \ifdefstring{\columns@position}{R}{\hfill}%
1903      }%
1904      \checkraw@text
1905      \checkverseL
1906      \checkverseR
1907      \checkpb@columns
1908      \repeat}

```

Having completed a pair of chunks, write the number of lines in each chunk to the respective section files. Increment pstart counters and reset line numbering if it's by pstart.

```

1909      \@writelinesinparL
1910      \@writelinesinparR
1911      \check@pstarts
1912      \ifbypstart@%
1913      \write\linenum@out{\string\@set[1]}
1914      \resetprevline@
1915      \fi
1916      \ifbypstart@R
1917      \write\linenum@outR{\string\@set[1]}
1918      \resetprevline@

```

```

1919      \fi
1920      \repeat

```

Having output all chunks, make sure all notes have been output, then zero counts ready for the next set of texts. The boolean tests for stanza are switched to false.

```

1921      \flush@notes
1922      \flush@notesR
1923      \endgroup

1924      \global\l@dpscL=\z@
1925      \global\l@dpscR=\z@
1926      \global\l@dnpstartsL=\z@
1927      \global\l@dnpstartsR=\z@
1928      \l@dprintingcolumnsfalse%
1929      \ignorespaces
1930      \global\instanzaLfalse
1931      \global\instanzaRfalse}
1932

```

`\print@columnseparator` `\print@columnseparator` prints the column separator, with surrounding spaces (as the user has set them). We use the \TeX `\ifdim` instead of `etoolbox` to avoid having `\hfill` in a `{}`, which deletes some space (but not much).

```

1933 \def\print@columnseparator{%
1934   \ifdim\beforecolumnseparator<0pt%
1935     \hfill%
1936   \else%
1937     \hspace{\beforecolumnseparator}%
1938   \fi%
1939   \columnseparator%
1940   \ifdim\aftercolumnseparator<0pt%
1941     \hfill%
1942   \else%
1943     \hspace{\beforecolumnseparator}%
1944   \fi%
1945 }%
1946 %\end{macrocode}
1947 % \end{macro}
1948 % \begin{macro}{\checkpb@columns}
1949 % \cs{checkpb@columns} prevent or make pagebreaking in columns, depending of the use of \
1950 %   \begin{macrocode}
1951
1952 \newcommand{\checkpb@columns}{%
1953   \newif\if@pb
1954   \newif\if@nopb
1955   \IfStrEq{\led@pb@setting}{before}{
1956     \numdef{\next@absline}{\the\absline@num+1}%
1957     \numdef{\next@abslineR}{\the\absline@numR+1}%
1958     \xifinlistcs{\next@absline}{l@prev@pb}{\@pbtrue}{}%
1959     \xifinlistcs{\next@abslineR}{l@prev@pbR}{\@pbtrue}{%
1960     \xifinlistcs{\next@absline}{l@prev@nopb}{\@nopbtrue}{}%

```

```

1961 \xifinlistcs{\next@abslineR}{l@prev@nopbR}{\@nopbtrue}{%
1962 }{}
1963 \IfStrEq{\led@pb@setting}{after}{%
1964 \xifinlistcs{\the\absline@num}{l@prev@pb}{\@pbtrue}{}%
1965 \xifinlistcs{\the\absline@numR}{l@prev@pbR}{\@pbtrue}{}%
1966 \xifinlistcs{\the\absline@num}{l@prev@nopb}{\@nopbtrue}{}%
1967 \xifinlistcs{\the\absline@numR}{l@prev@nopbR}{\@nopbtrue}{}%
1968 }{}
1969 \if@nopb\nopagebreak[4]\enlargethispage{\baselineskip}\fi
1970 \if@pb\pagebreak[4]\fi
1971 }

```

`\columnseparator` The separator between line pairs in parallel columns is in the form of a vertical rule extending a little below the baseline and with a height slightly greater than the `\baselineskip`. The width of the rule is `\columnrulewidth` (initially 0pt so the rule is invisible).

```

1972 \newcommand*{\columnseparator}{%
1973 \smash{\rule[-0.2\baselineskip]{\columnrulewidth}{1.05\baselineskip}}}
1974 \newdimen\columnrulewidth
1975 \columnrulewidth=\z@
1976

```

`\columnspan` The position of the `\Columns` in a page. Default value is R. Stored in `\columns@position`.

```

1977 \newcommand*{\columnspan}[1]{%
1978 \xdef\columns@position{#1}%
1979 }%
1980 \xdef\columns@position{R}%

```

`\beforecolumnseparator` `\aftercolumnseparator` and `\aftercolumnseparator` lengths are defined to -1pt. If user changes them to a positive length, the lengths are used to define blank spaces before / after the column separator, instead of `\hfill`.

```

1981 \newlength{\beforecolumnseparator}%
1982 \setlength{\beforecolumnseparator}{-2pt}%
1983
1984 \newlength{\aftercolumnseparator}%
1985 \setlength{\aftercolumnseparator}{-2pt}%
1986

```

`\setwidthliketwocolumns@L` The `\setwidth...` macros are called in `\beginnumbering` in a **non-parallel** typesetting context, to fix the width of the lines to be vertically aligned with parallel columns. They are also called at the beginning of a note's group, if some options are enabled. The `\setposition...` macros are called in `\beginnumbering` in a **non-parallel** typesetting context to fix the position of the lines. The `\setnoteposition...` macros are called in `\xxxfootstart` in a **non-parallel** typesetting context to fix the position of notes block.

```

1987 \newcommand{\setwidthliketwocolumns@L}{%
1988 % Temporary dimension, initially equal to the standard hsize, i.e. text width

```

```

1989 % \begin{macrocode}
1990 \newdimen\temp%
1991 \temp=\hsize%

      Hsize : Left + Right width
1992 \hsize=\Lcolwidth%
1993 \advance\hsize\Rcolwidth%

      Now, calculating the remaining space
1994 \advance\temp-\hsize%

      And multiply the hsize by 2/3 of this space
1995 \multiply\temp by 2%
1996 \divide\temp by 3%
1997 \advance\hsize\temp%
1998 }%
1999
2000 \newcommand{\setpositionliketwocolumns@L}{%
2001 \renewcommand{\ledrlfill}{\hfill}%
2002 }%
2003
2004 \newcommand{\setnotespositionliketwocolumns@L}{%
2005 }%
2006
2007
2008 \newcommand{\setwidthliketwocolumns@C}{%
2009 % Temporary dimension, initially equal to the standard hsize, i.e. text width
2010 \newdimen\temp%
2011 \temp=\hsize%
2012 % Hsize : Left + Right width
2013 \hsize=\Lcolwidth%
2014 \advance\hsize\Rcolwidth%
2015 % Now, calculating the remaining space
2016 \advance\temp-\hsize%

      And multiply the hsize by 1/2 of this space
2017 \divide\temp by 2%
2018 \advance\hsize\temp%
2019 }%
2020
2021 \newcommand{\setpositionliketwocolumns@C}{%
2022 \doinsidelinehook{\hfill}%
2023 \renewcommand{\ledrlfill}{\hfill}%
2024 }%
2025
2026 \newcommand{\setnotespositionliketwocolumns@C}{%
2027 \newdimen\temp%
2028 \newdimen\tempa%
2029 \temp=\hsize%

```

```

2030 \tempa=\Lcolwidth%
2031 \advance\tempa\Rcolwidth%
2032 \advance\temp-\tempa%
2033 \divide\temp by 2%
2034 \leftskip=\temp%
2035 \rightskip=-\temp%
2036 }%
2037
2038 \newcommand{\setwidthliketwocolumns@R}{%
    Temporary dimension, initially equal to the standard hsize, i.e. text width
2039 \newdimen\temp%
2040 \temp=\hsize%
    Hsize : Left + Right width
2041 \hsize=\Lcolwidth%
2042 \advance\hsize\Rcolwidth%
    Now, calculating the remaining space
2043 \advance\temp-\hsize%
    And multiply the hsize by 2/3 of this space
2044 \multiply\temp by 2%
2045 \divide\temp by 3%
2046 \advance\hsize\temp%
2047 }%
2048
2049 \newcommand{\setpositionliketwocolumns@R}{%
2050 \doinsidelinehook{\hfill}%
2051 }%
2052
2053 \newcommand{\setnotespositionliketwocolumns@R}{%
2054 \newdimen\temp%
2055 \newdimen\tempa%
2056 \temp=\hsize%
2057 \tempa=\Lcolwidth%
2058 \advance\tempa\Rcolwidth%
2059 \advance\temp-\tempa%
2060 \divide\temp by 2%
2061 \leftskip=\temp%
2062 \rightskip=-\temp%
2063 }%
2064

```

28 Parallel pages

This is considerably more complicated than parallel columns.

`\numpagelinesL` Counts for the number of lines on a left or right page, and the smaller of the
`\numpagelinesR` number of lines on a pair of facing pages.
`\l@dmminpagelines`

```

2065 \newcount\numpagelinesL
2066 \newcount\numpagelinesR
2067 \newcount\l@dminpagelines
2068

```

\Pages The **\Pages** command results in the previous Left and Right texts being typeset on matching facing pages. There should be equal numbers of chunks in the left and right texts.

```

2069 \newcommand*{\Pages}{%
2070   \l@dprintingpagetrue%
2071   \eledsection@correcting@skip=-2\baselineskip% line correcting for section titles.
2072   \parledgroup@notespacing@set@correction
2073   \typeout{}
2074   \typeout{***** PAGES *****}
2075   \ifnum\l@dnumpstartsL=\l@dnumpstartsR\else
2076     \led@err@BadLeftRightPstarts{\the\l@dnumpstartsL}{\the\l@dnumpstartsR}%
2077   \fi

```

As **\Pages** must be called outside of the pages environment, we have to redefine the **\Lcolwidth** and **\Rcolwidth** lengths, to prevent false overfull hboxes.

```

2078   \setlength{\Lcolwidth}{\textwidth}%
2079   \setlength{\Rcolwidth}{\textwidth}%

```

Get onto an empty even (left) page, then initialise counters, etc.

```

2080   \cleartol@devenpage
2081   \begingroup
2082     \l@dzeropenalties
2083     \endgraf\global\num@lines=\prevgraf
2084     \global\num@linesR=\prevgraf
2085     \global\par@line=\z@
2086     \global\par@lineR=\z@
2087     \global\l@dpscL=\z@
2088     \global\l@dpscR=\z@
2089     \writtenlinesLfalse
2090     \writtenlinesRfalse

```

The footnotes are printed in way which is different way from the one expected in **eledmac**, as we may want to have the notes printed in one side only.

```

2091     \let\print@Xnotes\print@Xnotes@forpages%
2092     \let\print@notesX\print@notesX@forpages%

```

Check if there are chunks to be processed.

```

2093     \check@pstarts
2094     \loop\if@pstarts

```

Loop over the number of chunks, incrementing the chunk counts (**\l@dpscL** and **\l@dpscR** are chunk (box) counts.)

```

2095       \global\advance\l@dpscL \@ne
2096       \global\advance\l@dpscR \@ne

```

Calculate the maximum number of real text lines in the chunk pair, storing the result in the relevant \l@dmaxlinesinpar.

```

2097      \getlinesfromparlistL
2098      \getlinesfromparlistR
2099      \l@dcalc@maxoftwo{\@cs@linesinparL}{\@cs@linesinparR}%
2100      {\usernamecount{\l@dmaxlinesinpar\the\l@dpscL}}%
2101      \check@pstarts
2102      \repeat

```

Zero the counts again, ready for the next bit.

```

2103      \global\l@dpscL=\z@
2104      \global\l@dpscR=\z@

```

Get the number of lines on the first pair of pages and store the minimum in \l@dminpagelines.

```

2105      \getlinesfrompagelistL
2106      \getlinesfrompagelistR
2107      \l@dcalc@minoftwo{\@cs@linesonpageL}{\@cs@linesonpageR}%
2108      {\l@dminpagelines}%

```

Now we start processing the left and right chunks (\l@dpscL and \l@dpscR count the left and right chunks), starting with the first pair.

```

2109      \check@pstarts
2110      \if@pstarts

```

Increment the chunk counts to get the first pair. Restore also the value of public pstart counters.

```

2111      \global\advance\l@dpscL \@ne
2112      \global\advance\l@dpscR \@ne
2113      \restore@pstartL@pc%
2114      \restore@pstartR@pc%

```

We haven't processed any lines from these chunks yet, so zero the respective line counts.

```

2115      \global\@donereallinesL=\z@
2116      \global\@donetotallinesL=\z@
2117      \global\@donereallinesR=\z@
2118      \global\@donetotallinesR=\z@

```

Start a loop over the boxes (chunks).

```

2119      \checkraw@text
2120 %      \begingroup
2121 {      \loop\ifaraw@text

```

See if there is more that can be done for the left page and set up the left language.

```

2122      \checkpageL
2123      \l@duselanguage{\theledlanguageL}%
2124 {      \loop\ifl@dsamepage%

```

Process the next (left) text line, adding it to the page. Eventually, adds the optional argument of pstart.

```

2125         \ifdefstring{\@eledsectnotoc}{L}{\ledsectnotoc}{%}
2126         \csuse{before@pstartL@the\l@dpscl}%
2127         \global\csundef{before@pstartL@the\l@dpscl}
2128         \do@lineL
2129         \xifinlist{\the\l@dpscl}{\eled@sections@}
2130         {\print@eledsectionL}%
2131         {}%
2132         \advance\numpagelinesL \@ne

```

When using shiftedpstarts option, a \l@dleftbox with a null height is not printed. That means we do not insert blank lines at the end of a left chunk lower than the corresponding right chunk. However, a \l@dleftbox with a null height will advance the \pagetotal in any case. Because if we do not do this, the \checkpageL could let \ifl@pagefull to false, and consequently a \@lopL equal to 1000 could be written in the numbered file, even if all the lines actually needed for the current page have been printed. \l@dleftbox

```

2133         \ifshiftedpstarts
2134             \ifdim\ht\l@dleftbox>Opt\hb@xt@%
2135                 \hsize{\ledstrutL\unhbox\l@dleftbox}%
2136             \else%
2137                 \dimen0=\pagetotal%
2138                 \advance\dimen0 by \baselineskip%
2139                 \global\pagetotal=\dimen0%
2140             \fi%
2141         \else%
2142             \parledgroup@correction@notespacing{L}
2143             \hb@xt@ \hsize{\ledstrutL\unhbox\l@dleftbox}%
2144         \fi

```

Perhaps we have to move to the next (left) box. Check if we have got all we can onto the page. If not, repeat for the next line. Check if we have to print the optional argument of the last pend. Check if the page is full. Check if the verse is split in two subsequent pages. Check there is any forced page breaks.

```

2145         \get@nextboxL%
2146         \ifprint@last@after@pendL%
2147             \csuse{after@pendL@the\l@dpscl}%
2148             \global\csundef{after@pendL@the\l@dpscl}%
2149         \fi%
2150         \checkpageL%
2151         \checkverseL
2152         \checkpbl
2153         \repeat

```

That (left) page has been filled. Output the number of real lines on the page — if the page break is because the page has been filled with lines, use the actual number, otherwise the page has been ended early in order to synchronise with the facing page so use an impossibly large number.


```

2154      \ifl@dpagfull
2155      \@writelinesonpageL{\the\numpagelinesL}%
2156    \else
2157      \@writelinesonpageL{1000}%
2158    \fi

```

Reset to zero the left-page line count, clear the page to get onto the facing (odd, right) page, and reinitialize the accumulated dimension of interline correction for notes in parallel ledgroup.

```

2159      \numpagelinesL \z@
2160      \parledgroup@correction@notespacing@init
2161      \clearl@dleftpage }%

```

Now do the same for the right text.

```

2162      \checkpageR%
2163      \l@duselanguage{\theledlanguageR}%
2164    {
2165      \loop\ifl@dsamepage%
2166      \initnumbering@sectcountR
2167      \ifdefstring{\@eledsectnotoc}{R}{\ledsectnotoc}{%
2168      \csuse{before@pstartR@the\l@dpscR}%
2169      \global\csundef{before@pstartR@the\l@dpscR}
2170      \do@lineR
2171      \xifinlist{\the\l@dpscR}{\eled@sectionsR@@}
2172      {\print@eledsectionR}%
2173      }%
2174      \advance\numpagelinesR \@ne
2175      \ifshiftedpstarts
2176        \ifdim\ht\l@drightbox>0pt\hb@xt@%
2177          \hsize{\ledstrutR\unhbox\l@drightbox}%
2178        \else%
2179          \dimen0=\pagetotal%
2180          \advance\dimen0 by \baselineskip%
2181          \global\pagetotal=\dimen0%
2182        \fi%
2183      \else%
2184        \parledgroup@correction@notespacing{R}
2185        \hb@xt@ \hsize{\ledstrutR\unhbox\l@drightbox}%
2186      \fi
2187      \get@nextboxR%
2188      \ifprint@last@after@pendR%
2189        \csuse{after@pendR@the\l@dpscR}%
2190        \global\csundef{after@pendR@the\l@dpscR}%
2191      \fi%
2192      \checkpageR%
2193      \checkverseR
2194      \checkpbR
2195      \repeat
2196      \ifl@dpagfull
2197        \@writelinesonpageR{\the\numpagelinesR}%
2198      \else

```

```

2198         \@writelinesonpageR{1000}%
2199         \fi
2200         \numpagelinesR=\z@
2201         \parledgroup@correction@notespacing@init
    The page is full, so move onto the next (left, odd) page and repeat left text
    processing.
2202         \clearl@dtrightpage}
    More to do? If there is we have to get the number of lines for the next pair of
    pages before starting to output them.
2203         \checkraw@text
2204         \ifaraw@text
2205             \getlinesfrompagelistL
2206             \getlinesfrompagelistR
2207             \l@dcalc@minoftwo{\@cs@linesonpageL}{\@cs@linesonpageR}%
2208                                 {\l@dminpagelines}%
2209         \fi
2210         \repeat}
    We have now output the text from all the chunks.
2211     \fi
    Make sure that there are no inserts hanging around.
2212     \flush@notes
2213     \flush@notesR
2214     \endgroup
    Zero counts ready for the next set of left/right text chunks. The boolean tests for
    stanza are switched to false.
2215     \global\l@dpscL=\z@
2216     \global\l@dpscR=\z@
2217     \global\l@dnumpststartsL=\z@
2218     \global\l@dnumpststartsR=\z@
2219     \global\instanzaLfalse
2220     \global\instanzaRfalse
2221     \l@dprintingpagesfalse%
2222     \finish@Pages@notes%Needed to prevent final notes overlap line number
2223     \ignorespaces}
2224

```

\finish@Pages@notes This macro ensures that all long notes are printed at the end of **\Pages** typesetting, and that there is no more long notes left for the next pages.

```

2225 \newcommand{\finish@Pages@notes}{%
2226   \def\do##1{%
2227     \ifvoid\csuse{##1footins}%
2228       \ifvoid\csuse{footins##1}\else%
2229         \newpage\null%
2230         \listbreak%
2231       \fi%
2232     \else%

```

```

2233      \newpage\null%
2234      \listbreak%
2235      \fi%
2236    }%
2237    \dolistloop{\@series}%
2238  }%

```

`\ledstrutL` Struts inserted into leftand right text lines.

```

\ledstrutR 2239 \newcommand*{\ledstrutL}{\strut}
          2240 \newcommand*{\ledstrutR}{\strut}
          2241

```

`\cleartoevenpage` `\cleartoevenpage`, which is defined in the memoir class, is like `\clear(double)page` except that we end up on an even page. `\cleartol@devenpage` is similar except that it first checks to see if it is already on an empty page.

```

2242 \providecommand{\cleartoevenpage}[1][\@empty]{%
2243   \clearpage
2244   \ifodd\c@page\hbox{ }#1\clearpage\fi}
2245 \newcommand*{\cleartol@devenpage}{%
2246   \ifdim\pagetotal<\topskip% on an empty page
2247   \else
2248     \clearpage
2249   \fi
2250   \ifodd\c@page\hbox{ }\clearpage\fi}

```

`\clearl@dleftpage` `\clearl@dleftpage` and `\clearl@drighthouse` get us onto an odd and even page, respectively, checking that we end up on the subsequent page. Both commands use `\newpage` and not `\clearpage`. Because `\clearpage` prints all footnotes before the next page, even if it has to add new empty pages, while `\newpage` does not. And as we want notes started in the left page continue in the right page and *vice-versa*, we must use `\newpage` and not `\clearpage`

```

2251 \newcommand*{\clearl@dleftpage}{%
2252   \ifdim\pagetotal=0pt\hbox{ }\fi%
2253   \newpage%
2254   \ifodd\c@page\else
2255     \led@err@LeftOnRightPage
2256     \hbox{ }%
2257     \cleardoublepage
2258   \fi}
2259
2260 \newcommand*{\clearl@drighthouse}{%
2261   \ifdim\pagetotal=0pt\hbox{ }\fi%
2262   \newpage%
2263   \ifodd\c@page
2264     \led@err@RightOnLeftPage
2265     \hbox{ }%
2266     \cleartoevenpage
2267   \fi}
2268

```

`\getlinesfromparlistL` `\getlinesfromparlistL` gets the next entry from the `\linesinpar@listL` and `\@cs@linesinparL` puts it into `\@cs@linesinparL`; if the list is empty, it sets `\@cs@linesinparL` to `\getlinesfromparlistR` 0. Similarly for `\getlinesfromparlistR`.

```
\@cs@linesinparR 2269 \newcommand*{\getlinesfromparlistL}{%
2270   \ifx\linesinpar@listL\empty
2271     \gdef\@cs@linesinparL{0}%
2272   \else
2273     \gl@p\linesinpar@listL\to\@cs@linesinparL
2274   \fi}
2275 \newcommand*{\getlinesfromparlistR}{%
2276   \ifx\linesinpar@listR\empty
2277     \gdef\@cs@linesinparR{0}%
2278   \else
2279     \gl@p\linesinpar@listR\to\@cs@linesinparR
2280   \fi}
2281
```

`\getlinesfrompagelistL` `\getlinesfrompagelistL` gets the next entry from the `\linesonpage@listL` and `\@cs@linesonpageL` puts it into `\@cs@linesonpageL`; if the list is empty, it sets `\@cs@linesonpageL` `\getlinesfrompagelistR` to 1000. Similarly for `\getlinesfrompagelistR`.

```
\@cs@linesonpageR 2282 \newcommand*{\getlinesfrompagelistL}{%
2283   \ifx\linesonpage@listL\empty
2284     \gdef\@cs@linesonpageL{1000}%
2285   \else
2286     \gl@p\linesonpage@listL\to\@cs@linesonpageL
2287   \fi}
2288 \newcommand*{\getlinesfrompagelistR}{%
2289   \ifx\linesonpage@listR\empty
2290     \gdef\@cs@linesonpageR{1000}%
2291   \else
2292     \gl@p\linesonpage@listR\to\@cs@linesonpageR
2293   \fi}
2294
```

`\@writelinesonpageL` These macros output the number of lines on a page to the section file in the form of `\@lopL` or `\@lopR` macros.

```
2295 \newcommand*{\@writelinesonpageL}[1]{%
2296   \edef\next{\write\linenum@out{\string\@lopL{#1}}}%
2297   \next}
2298 \newcommand*{\@writelinesonpageR}[1]{%
2299   \edef\next{\write\linenum@outR{\string\@lopR{#1}}}%
2300   \next}
2301
```

`\l@dcalc@maxoftwo` `\l@dcalc@maxoftwo{<num>}{<num>}{<count>}` sets `<count>` to the maximum of the two `<num>`.

Similarly `\l@dcalc@minoftwo{<num>}{<num>}{<count>}` sets `<count>` to the minimum of the two `<num>`.

```

2302 \newcommand*{\l@dcalc@maxoftwo}[3]{%
2303   \ifnum #2>#1\relax
2304     #3=#2\relax
2305   \else
2306     #3=#1\relax
2307   \fi}
2308 \newcommand*{\l@dcalc@minoftwo}[3]{%
2309   \ifnum #2<#1\relax
2310     #3=#2\relax
2311   \else
2312     #3=#1\relax
2313   \fi}
2314

```

\ifl@dsamepage \checkpageL tests if the space and lines already taken on the page by text and foot-
 \l@dsamepagetrue notes is less than the constraints. If so, then \ifl@dpagetrue is set FALSE and
 \l@dsamepagefalse \ifl@dsamepage is set TRUE. If the page is spatially full then \ifl@dpagetrue
 \ifl@dpagetrue is set TRUE and \ifl@dsamepage is set FALSE. If it is not spatially full but
 \l@dpagetrue the maximum number of lines have been output then both \ifl@dpagetrue and
 \l@dpagetruefalse \ifl@dsamepage are set FALSE.

```

\checkpageL 2315 \newif\ifl@dsamepage
\checkpageR 2316 \l@dsamepagetrue
2317 \newif\ifl@dpagetrue
2318
2319 \newcommand*{\checkpageL}{%
2320   \l@dpagetrue
2321   \l@dsamepagetrue
2322   \check@goal
2323   \ifdim\pagetotal<\ledthegoal
2324     \ifnum\l@dpagetrue<\l@dsamepagetrue
2325       \else
2326         \l@dsamepagefalse
2327         \l@dpagetruefalse
2328       \fi
2329     \else
2330       \l@dsamepagefalse
2331       \l@dpagetrue
2332     \fi%
2333   \ifprint@last@after@pendL%
2334     \l@dpagetruefalse%
2335     \l@dsamepagefalse%
2336   \print@last@after@pendLfalse%
2337   \fi%
2338 }%
2339
2340 \newcommand*{\checkpageR}{%
2341   \l@dpagetrue
2342   \l@dsamepagetrue
2343   \check@goal

```

```

2344 \ifdim\pagetotal<\ledthegoal
2345 \ifnum\numpagelinesR<\l@dmnpagelines
2346 \else
2347 \l@dsamepagefalse
2348 \l@dpagfullfalse
2349 \fi
2350 \else
2351 \l@dsamepagefalse
2352 \l@dpagfulltrue
2353 \fi%
2354 \ifprint@last@after@pendR%
2355 \l@dpagfullfalse%
2356 \l@dsamepagefalse%
2357 \print@last@after@pendRfalse%
2358 \fi%
2359 }%
2360

```

`\checkpbL` `\checkpbL` and `\checkpbR` are called after each line is printed, and after the page is checked. These commands correct page breaks depending on `\ledpb` and `\lednopb`.

```

2361 \newcommand{\checkpbL}{
2362 \IfStrEq{\led@pb@setting}{after}{
2363 \xifinlistcs{\the\absline@num}{\l@prev@pb}{\l@dpagfulltrue\l@dsamepagefalse}{
2364 \xifinlistcs{\the\absline@num}{\l@prev@nopb}{\l@dpagfullfalse\l@dsamepagetrue}{
2365 }{
2366 \IfStrEq{\led@pb@setting}{before}{
2367 \numdef{\next@absline}{\the\absline@num+1}
2368 \xifinlistcs{\next@absline}{\l@prev@pb}{\l@dpagfulltrue\l@dsamepagefalse}{
2369 \xifinlistcs{\next@absline}{\l@prev@nopb}{\l@dpagfullfalse\l@dsamepagetrue}{
2370 }{
2371 }
2372 }
2373 \newcommand{\checkpbR}{
2374 \IfStrEq{\led@pb@setting}{after}{
2375 \xifinlistcs{\the\absline@numR}{\l@prev@pbR}{\l@dpagfulltrue\l@dsamepagefalse}{
2376 \xifinlistcs{\the\absline@numR}{\l@prev@nopbR}{\l@dpagfullfalse\l@dsamepagetrue}{
2377 }{
2378 \IfStrEq{\led@pb@setting}{before}{
2379 \numdef{\next@abslineR}{\the\absline@numR+1}
2380 \xifinlistcs{\next@abslineR}{\l@prev@pbR}{\l@dpagfulltrue\l@dsamepagefalse}{
2381 \xifinlistcs{\next@abslineR}{\l@prev@nopbR}{\l@dpagfullfalse\l@dsamepagetrue}{
2382 }{
2383 }

```

`\checkverseL` `\checkverseL` and `\checkverseR` are called after each line is printed. They prevent page break inside verse.

```

2384 \newcommand{\checkverseL}{
2385 \ifinstanzaL

```

```

2386 \iflednopbinverse
2387 \ifinserthangingsymbol
2388 \numgdef{\prev@abslineinverse}{\the\absline@num-1}
2389 \IfStrEq{\led@pb@setting}{after}{\lednopbnum{\prev@abslineinverse}}{}
2390 \IfStrEq{\led@pb@setting}{before}{\ifnum\numpagelinesL<3\ledpbnum{\prev@abslineinverse}\fi}{}
2391 \fi
2392 \fi
2393 \fi
2394 }
2395 \newcommand{\checkverseR}{
2396 \ifinstanzaR
2397 \iflednopbinverse
2398 \ifinserthangingsymbolR
2399 \numgdef{\prev@abslineinverse}{\the\absline@numR-1}
2400 \IfStrEq{\led@pb@setting}{after}{\lednopbnumR{\prev@abslineinverse}}{}
2401 \IfStrEq{\led@pb@setting}{before}{\ifnum\numpagelinesR<3\ledpbnumR{\prev@abslineinverse}\fi}{}
2402 \fi
2403 \fi
2404 \fi
2405 }

```

\ledthegoal \ledthegoal is the amount of space allowed to taken by text and footnotes on
\goalfraction a page before a forced pagebreak. This can be controlled via \goalfraction.
\check@goal \ledthegoal is calculated via \check@goal.

```

2406 \newdimen\ledthegoal
2407 \ifshiftedpstarts
2408 \newcommand*{\goalfraction}{0.95}
2409 \else
2410 \newcommand*{\goalfraction}{0.9}
2411 \fi
2412
2413 \newcommand*{\check@goal}{%
2414 \ledthegoal=\goalfraction\pagegoal}
2415

```

\ifwrittenlinesL Booleans for whether line data has been written to the section file.

```

\ifwrittenlinesL 2416 \newif\ifwrittenlinesL
2417 \newif\ifwrittenlinesR
2418

```

\get@nextboxL If the current box is not empty (i.e., still contains some lines) nothing is done.
\get@nextboxR Otherwise if and only if a synchronisation point is reached the next box is started.

```

2419 \newcommand*{\get@nextboxL}{%
2420 \ifvbox\namebox{1@dLcolrawbox\the\l@dpscl}% box is not empty

The current box is not empty; do nothing.

2421 \else% box is empty

The box is empty. Check if enough lines (real and blank) have been output.

```

```

2422     \ifnum\usernamecount{l@dmxlinesinpar\the\l@dpscL}>\@donetotallinesL
2423     \parledgroup@notes@endL
2424     \else
    Sufficient lines have been output.
2425     \ifnum\usernamecount{l@dmxlinesinpar\the\l@dpscL}=\@donetotallinesL
2426     \parledgroup@notes@endL
2427     \fi
2428     \ifwrittenlinesL\else
    Write out the number of lines done, and set the boolean so this is only done once.
2429     \@writelinesinparL
2430     \writtenlinesLtrue
2431     \fi
2432     \ifnum\l@dnumstartsL>\l@dpscL
    There are still unprocessed boxes. Recalculate the maximum number of lines
    needed, and move onto the next box (by incrementing \l@dpscL). If needed, restart
    the line numbering.
2433     \writtenlinesLfalse
2434     \ifbypstart@
2435     \global\line@num=0%
2436     \resetprevline@%
2437     \fi
2438 % Add the content of the optional argument of the previous \cs{pend}.
2439 % \begin{macrocode}
2440     \csuse{after@pendL@the\l@dpscL}%
2441     \global\csundef{after@pendL@the\l@dpscL}%
    Check the number of lines
2442     \l@dcalc@maxoftwo{\the\usernamecount{l@dmxlinesinpar\the\l@dpscL}}%
2443     {\the\@donetotallinesL}%
2444     {\usernamecount{l@dmxlinesinpar\the\l@dpscL}}%
2445     \global\@donetotallinesL \z@
    Go to the next pstart
2446     \global\advance\l@dpscL \@ne
2447     \global\pstartnumtrue%
2448     \restore@pstartL@pc%
    Add notes of parallel ledgroup.
2449     \parledgroup@notes@endL
2450     \parledgroup@correction@notes@spacing@final{L}
2451     \else
    Add the content of the optional argument of the last \pend.
2452     \print@last@after@pendLtrue%
2453     \fi
2454     \fi
2455     \fi}

```



```

2456 \newcommand*{\get@nextboxR}{%
2457   \ifvbox\namebox{1@dRcolrawbox\the\l@dpscR}% box is not empty
2458   \else% box is empty
2459     \ifnum\usernamecount{1@dmaxlinesinpar\the\l@dpscR}>\@donetotallinesR
2460       \parledgroup@notes@endR
2461     \else
2462       \ifnum\usernamecount{1@dmaxlinesinpar\the\l@dpscR}=\@donetotallinesR
2463         \parledgroup@notes@endR
2464       \fi
2465     \ifwrittenlinesR\else
2466       \@writelinesinparR
2467       \writtenlinesRtrue
2468     \fi
2469     \ifnum\l@dnumpstartsR>\l@dpscR
2470       \writtenlinesRfalse
2471       \ifbypstart@R
2472         \global\line@numR=0%
2473         \resetprevline@%
2474       \fi
2475       \csuse{after@pendR@the\l@dpscR}%
2476       \global\csundef{after@pendR@the\l@dpscR}%
2477       \l@dcalc@maxoftwo{\the\usernamecount{1@dmaxlinesinpar\the\l@dpscR}}%
2478         {\the\@donetotallinesR}%
2479       {\usernamecount{1@dmaxlinesinpar\the\l@dpscR}}}%
2480       \global\@donetotallinesR \z@
2481       \global\advance\l@dpscR \@ne
2482       \global\pstartnumRtrue%
2483       \restore@pstartR@pc%
2484       \parledgroup@notes@endR
2485       \parledgroup@correction@notes@spacing@final{R}
2486     \else
2487       \print@last@after@pendRtrue%
2488     \fi
2489   \fi
2490 \fi}
2491

```

29 Sections' titles' commands

`\eledsectnotoc` `\eledsectnotoc` just saves its content `\@eledsectnotoc`, which will be tested where sectioning commands will be printed.

```

2492 \newcommand{\eledsectnotoc}[1]{\xdef\@eledsectnotoc{#1}}
2493 \eledsectnotoc{R}

```

`\eledsectmark` `\eledsectmark` just saves its content `\@eledsectmark`, which will be tested where sectioning commands will be printed.

```

2494 \newcommand{\eledsectmark}[1]{\xdef\@eledsectmark{#1}}
2495 \eledsectmark{L}

```

`\eledsection@correcting@skip` Because the vertical correction needed after inserting a title in parallel depends whether we are in parallel columns or parallel pages, we stock its length in `\eledsection@correcting@skip`.

2496 `\newskip\eledsection@correcting@skip`

`\eled@sectioningR@out` We save the sectioning commands of the right side in the `\eled@sectioningR@out` file.

2497 `\newwrite\eled@sectioningR@out`

30 Page break/no page break, depending on the specific line

We need to adapt the macro of the homonym section of `eledmac` to `eledpar`.

`\prev@pbR` The `\l@prev@pbR` macro is a etoolbox list, which contains the lines in which page breaks occur (before or after). The `\l@prev@nopbR` macro is a etoolbox list, which contains the lines in which NO page breaks occur (before or after).

2498 `\def\l@prev@pbR{}`
 2499 `\def\l@prev@nopbR{}`

`\ledpbR` The `\ledpbR` macro writes the call to `\led@pbR` in line-list file. The `\ledpbnumR` macro writes the call to `\led@pbnumR` in line-list file. The `\lednopbR` macro writes the call to `\led@nopbR` in line-list file. The `\lednopbnumR` macro writes the call to `\led@nopbnumR` in line-list file.

2500 `\newcommand{\ledpbR}{\write\linenum@outR{\string\led@pbR}}`
 2501 `\newcommand{\ledpbnumR}[1]{\write\linenum@outR{\string\led@pbnumR{#1}}}`
 2502 `\newcommand{\lednopbR}{\write\linenum@outR{\string\led@nopbR}}`
 2503 `\newcommand{\lednopbnumR}[1]{\write\linenum@outR{\string\led@nopbnumR{#1}}}`

`\led@pbR` The `\led@pbR` add the absolute line number in the `\prev@pbR` list. The `\led@pbnumR` add the argument in the `\prev@pbR` list. The `\led@nopbR` add the absolute line number in the `\prev@nopbR` list. The `\led@nopbnumR` add the argument in the `\prev@nopbR` list.

2504 `\newcommand{\led@pbR}{\listxadd{\l@prev@pbR}{\the\absline@numR}}`
 2505 `\newcommand{\led@pbnumR}[1]{\listxadd{\l@prev@pbR}{#1}}`
 2506 `\newcommand{\led@nopbR}{\listxadd{\l@prev@nopbR}{\the\absline@numR}}`
 2507 `\newcommand{\led@nopbnumR}[1]{\listxadd{\l@prev@nopbR}{#1}}`

31 Parallel ledgroup

`\parledgroup@` The marks `\parledgroup` contains information about the beginnings and endings of notes in a parallel ledgroup. `\parledgroupseries@` contains the footnote series.

`\parledgrouptype@` `\parledgroupseries` contains the type of the footnote: critical (Xfootnote) or familiar (footnoteX).

2508 `\newmarks\parledgroup@`

```

2509 \newmarks\parledgroup@series
2510 \newmarks\parledgroup@type

```

`\parledgroup@notes@startL` `\parledgroup@notes@startL` and `\parledgroup@notes@startR` are used to mark the beginning of a note series in a parallel ledgroup.

```

2511 \newcommand{\parledgroup@notes@startL}{%
2512   \ifnum\usenamecount{1@dmaxlinesinpar\the\1@dpscL}>0%
2513     \IfStrEq{\splitfirstmarks\parledgroup@type}{footnoteX}{\csuse{bhooknoteX@\splitfirstmarks\parledg
2514     \IfStrEq{\splitfirstmarks\parledgroup@type}{Xfootnote}{\csuse{bhookXnote@\splitfirstmarks\parledg
2515   \fi%
2516   \global\ledgroupnotesL=true%
2517   \insert@noterule@ledgroup{L}%
2518 }
2519 \newcommand{\parledgroup@notes@startR}{%
2520   \ifnum\usenamecount{1@dmaxlinesinpar\the\1@dpscR}>0%
2521     \IfStrEq{\splitfirstmarks\parledgroup@type}{footnoteX}{\csuse{bhooknoteX@\splitfirstmarks\parledg
2522     \IfStrEq{\splitfirstmarks\parledgroup@type}{Xfootnote}{\csuse{bhookXnote@\splitfirstmarks\parledg
2523   \fi%
2524   \global\ledgroupnotesR=true%
2525   \insert@noterule@ledgroup{R}%
2526 }

```

`\parledgroup@notes@startL` `\parledgroup@notes@endL` and `\parledgroup@notes@endR` are used to mark the end of a note series in a parallel ledgroup.

```

2527 \newcommand{\parledgroup@notes@endL}{%
2528   \global\ledgroupnotesL=false%
2529 }
2530 \newcommand{\parledgroup@notes@endR}{%
2531   \global\ledgroupnotesR=false%
2532 }

```

`\insert@noterule@ledgroup` A `\vskip` is not used when the boxes are constructed. So we insert it before ledgroup note series when paralling lines are constructed. This is the goal of `\insert@noterule@ledgroup`

```

2533 \newcommand{\insert@noterule@ledgroup}[1]{
2534   \IfStrEq{\splitbotmarks\parledgroup@}{begin}{%
2535     \IfStrEq{\splitbotmarks\parledgroup@type}{Xfootnote}{
2536       \csuse{ifledgroupnotes#1@}
2537       \vskip\skip\csuse{mp\splitbotmarks\parledgroup@series footins}
2538       \csuse{\splitbotmarks\parledgroup@series footnoterule}
2539     \fi
2540   }
2541   {}
2542   \IfStrEq{\splitbotmarks\parledgroup@type}{footnoteX}{
2543     \csuse{ifledgroupnotes#1@}
2544     \vskip\skip\csuse{mpfootins\splitbotmarks\parledgroup@series}
2545     \csuse{footnoterule\splitbotmarks\parledgroup@series}
2546   \fi
2547   }{}

```

```

2548     }
2549     {}
2550 }

```

`\parledgroupnotespacing` `\parledgroupnotespacing` can be redefined by the user to change the interline spacing of ledgroup notes.

```

2551 \newcommand{\parledgroupnotespacing}{}

```

`\parledgroup@notespacing@correction` `\parledgroup@notespacing@correction` is the difference between a normal line skip and a line skip in a note. It's set by `\parledgroup@notespacing@set@correction`, called at the beginning of `\Pages`.

```

2552 \dimdef{\parledgroup@notespacing@correction}{0pt}
2553 \newcommand{\parledgroup@notespacing@set@correction}{%
2554   {\notefontsetup\parledgroupnotespacing\dimgdef{\temp@spacing}{\baselineskip}}%
2555   \dimgdef{\parledgroup@notespacing@correction}{\baselineskip-\temp@spacing}%
2556 }

```

`\parledgroup@correction@notespacing@init` `\parledgroup@correction@notespacing@init` sets the value of accumulated corrections of note spacing to 0 pt. It's called at the beginning of each pages AND at the end of each ledgroup.

```

2557 \newcommand{\parledgroup@correction@notespacing@init}{
2558   \dimdef{\parledgroup@notespacing@correction@accumulated}{0pt}
2559   \dimdef{\parledgroup@notespacing@correction@modulo}{0pt}
2560 }
2561 \parledgroup@correction@notespacing@init

```

`\parledgroup@correction@notespacing@final` `\parledgroup@correction@notespacing@final` adds the total space deleted because of correction for notes, in a parallel ledgroup. It also adds the space needed by the other side spaces between note rules and notes. It's called after the print of each pstart/pend.

```

2562 \newcommand{\parledgroup@correction@notespacing@final}[1]{
2563   \ifparledgroup
2564     \vspace{\parledgroup@notespacing@correction@accumulated}
2565     \parledgroup@correction@notespacing@init%
2566     \ifstrequal{#1}{L}{
2567       \numdef{\@checking}{\the\l@dpscL-1}
2568     }{
2569       \numdef{\@checking}{\the\l@dpscR-1}
2570     }
2571     \dimdef{\@beforenotes@current@diff}{\csuse{\parledgroup@beforenotes@\@checking L}-\cs
2572     \ifstrequal{#1}{L}%
2573       {% Left
2574         \ifdimgreater{\@beforenotes@current@diff}{0pt}{\vspace{-\@beforenotes@current@dif
2575       }%
2576       {% Right
2577         \ifdimgreater{\@beforenotes@current@diff}{0pt}{\vspace{\@beforenotes@current@diff}}-
2578       }%
2579     \fi
2580 }

```

`\parledgroup@correction@notespacing` `\parledgroup@correction@notespacing` is used before each printed line. If it's a line of notes in parallel ledgroup, the space `\parledgroup@notespacing@correction` is decreased, to make interline space correct. The decreased space is added to `\parledgroup@notespacing@correction@accumulated` and `\parledgroup@notespacing@correction@modulo`. If `\parledgroup@notespacing@correction@modulo` is equal or greater than `\baselineskip`:

- It is decreased by `\baselineskip`.
- The total of line number in the current page is decreased by one.

For example, suppose an normal interline of 24 pt and interline for note of 12 pt. That means that the two lines of notes take the place of one normal line. For every two lines of notes, the line total for the current place is decreased by one.

```

2581 {}
2582 \newcommand{\parledgroup@correction@notespacing}[1]{%
2583     \csuse{ifledgroupnotes#1@}%
2584     \vspace{-\parledgroup@notespacing@correction}%
2585     \dimdef{\parledgroup@notespacing@correction@accumulated}{\parledgroup@notespacing@correction@ac
2586     \dimdef{\parledgroup@notespacing@correction@modulo}{\parledgroup@notespacing@correction@modulo+
2587     \ifdimless{\parledgroup@notespacing@correction@modulo}{\baselineskip}{\advance\numpagelinesL
2588     \dimdef{\parledgroup@notespacing@correction@modulo}{\parledgroup@notespacing@correction@modulo-
2589     }% mean greater than equal
2590     \fi%
2591 }
```

`\parledgroup@beforenotesL` `\parledgroup@beforenotesL` and `\parledgroup@beforenotesR` store the total of space before notes in the current parallel ledgroup.

```

2592 \dimdef\parledgroup@beforenotesL{0pt}
2593 \dimdef\parledgroup@beforenotesR{0pt}
```

`\parledgroup@beforenotes@save` The macro `\parledgroup@beforenotes@save` dumps the space before notes of the current parallel ledgroup in a macro named with the current `pstart` number.

```

2594 \newcommand{\parledgroup@beforenotes@save}[1]{
2595     \ifparledgroup
2596     \csdimgdef{\parledgroup@beforenotes@the\csuse{lednumstarts#1}#1}{\csuse{\parledgroup@beforenotes
2597     \csdimgdef{\parledgroup@beforenotes#1}{0pt}
2598     \fi
2599 }
```

32 The End

`i/code`

Appendix A Some things to do when changing version

Appendix A.1 Migration to eledpar 1.4.3

Version 1.4.3 corrects a bug added in version 0.12, which made hanging verse automatically flush right, despite the given value of the first element of the `\setstanzaindents` command.

If, however, you want to return to automatic flush-right margins for verses with hanging indents, you have to redefine the `\hangingsymbol` command.

```
\renewcommand{\hangingsymbol}{\protect\hfill}
```

See the two following examples:

With standard `\hangingsymbol`:

A very long verse should be sometime hanged. The position of the hanging verse is fixed.

With the modification of `\hangingsymbol`:

A very long verse should sometimes be hanging. And we can see that an hanging verse is flush right.

References

- [LW90] John Lavagnino and Dominik Wujastyk. ‘An overview of EDMAC: a PLAIN TeX format for critical editions’. *TUGboat*, **11**, 4, pp. 623–643, November 1990. (Code available from CTAN in `macros/plain/contrib/edmac`)
- [Wil02] Peter Wilson. *The memoir class for configurable typesetting*. November 2002. (Available from CTAN in `macros/latex/contrib/memoir`)
- [Wil04] Peter Wilson and Maïeul Rouquette. *eledmac A presumptuous attempt to port EDMAC, TABMAC and EDSTANZA to LaTeX*. December 2004. (Available from CTAN in `macros/latex/contrib/eledmac`)

Index

Numbers written in *italic* refer to the page where the corresponding entry is described; numbers underlined refer to the code line of the definition; numbers in roman refer to the code lines where the entry is used.

Symbols	<code>\@adv</code>	361, 621, 622
<code>\&</code>	1660, 1661, 1665, 1681, 1704	<code>\@afterindentfalse</code> 764
<code>\@RTLtrue</code>	1031	<code>\@arabic</code> 213, 214, 812, 814

<code>\@astanza@line</code>	1680, 1683, <u>1691</u>	<code>\@nl@reg</code>	332
<code>\@auxout</code>	1768	<code>\@nl@regR</code>	<u>283</u>
<code>\@beforenotes@current@diff</code>		<code>\@nobreakfalse</code>	820, 868, 1024
	2571, 2574, 2577	<code>\@nobreaktrue</code>	818, 822, 866, 870, 1024
<code>\@chapter</code>	765	<code>\@nopbtrue</code>	1960, 1961, 1966, 1967
<code>\@checking</code>	2567, 2569, 2571	<code>\@oldnobreak</code>	818, 820, 866, 868, 923, 945	
<code>\@cs@linesinparL</code>	2099, <u>2269</u>	<code>\@pbtrue</code>	1958, 1959, 1964, 1965
<code>\@cs@linesinparR</code>	2099, <u>2269</u>	<code>\@pend</code>	<u>570</u> , 1835
<code>\@cs@linesonpageL</code>	..	2107, 2207, <u>2282</u>	<code>\@pendR</code>	<u>570</u> , 1840
<code>\@cs@linesonpageR</code>	..	2107, 2207, <u>2282</u>	<code>\@pstartsfalse</code>	<u>1809</u>
<code>\@currentlabel</code>	858, 905	<code>\@pstartstrue</code>	<u>1809</u>
<code>\@donereallinesL</code>		<code>\@cref</code>	<u>540</u>
	<u>966</u> , 996, 1835, 1837, 2115	<code>\@cref@reg</code>	568
<code>\@donereallinesR</code>		<code>\@schapter</code>	765
	<u>966</u> , 1084, 1840, 1842, 2117	<code>\@series</code>	710, 2237
<code>\@donetotallinesL</code>	<u>966</u> , 997,	<code>\@set</code>	<u>393</u> , 628, 629, 1913, 1917
		1000, 2116, 2422, 2425, 2443, 2445	<code>\@startstanza</code>	777, 778, 800, 801
<code>\@donetotallinesR</code>	<u>966</u> , 1085,	<code>\@stopastanza</code>	1681, <u>1686</u>
		1088, 2118, 2459, 2462, 2478, 2480	<code>\@sw</code>	555
<code>\@eled@sectioningfalse</code>	1035	<code>\@tag</code>	1589, 1591, 1595
<code>\@eled@sectioningtrue</code>	1025	<code>\@temp</code>	727, 728, 733, 734
<code>\@eledsectionL</code>	..	<u>1844</u> , 1878, 1894, 1895	<code>\@templ@d</code>	1554, 1556
<code>\@eledsectionR</code>	..	<u>1844</u> , 1889, 1899, 1900	<code>\@templ@n</code>	1555, 1556
<code>\@eledsectmark</code>	..	1022, 1874, 1885, 2494	<code>\@tmp</code>	687, 689, 690, 693–695
<code>\@eledsectnotoc</code>		1021, 2125, 2166, 2492	<code>\@tmpa</code>	688, 689, 694, 697
<code>\@gobble</code>	550	<code>\@writelinesinparL</code>	..	<u>1833</u> , 1909, 2429
<code>\@gobbletwo</code>	555	<code>\@writelinesinparR</code>	..	<u>1833</u> , 1910, 2466
<code>\@insertR</code>	1390–1392, 1405–1407	<code>\@writelinesonpageL</code>	..	2155, 2157, <u>2295</u>
<code>\@l@dttempcnta</code>	434,	<code>\@writelinesonpageR</code>	..	2196, 2198, <u>2295</u>
		436, 438, 439, 443, 445, 447,	<code>\@xloop</code>	1403
		448, 1142, 1181, 1182, 1184,			
		1186, 1189, 1190, 1207–1211,			
		1213, 1220, 1225, 1229, 1237,			
		1242, 1246, 1279, 1282, 1284, 1288			
<code>\@l@dttempcntb</code>	169, 171, 173,			
		1172, 1173, 1220, 1225, 1229,			
		1237, 1242, 1246, 1271, 1275,			
		1288, 1296–1298, 1300, 1321–			
		1323, 1325, 1342–1344, 1346,			
		1535, 1536, 1564–1566, 1568,			
		1732–1736, 1740–1743, 1747–1750			
<code>\@lab</code>	554, <u>1522</u>			
<code>\@lock</code>	984, 1122			
<code>\@lockR</code>	..	49, 305, 307, 309, 322, 468,			
		484, 485, 487, 488, 516, 517,			
		519, 1071, 1104, 1148, 1150,			
		1151, 1153, 1234, 1251, 1253, 1255			
<code>\@lopL</code>	<u>579</u> , 2296			
<code>\@lopR</code>	550, <u>579</u> , 2299			
<code>\@nl</code>	283, 604, 606			

- `\add@inserts@nextR` 1378
`\add@insertsR` 1080, 1378
`\add@penaltiesL` 995, 1399
`\add@penaltiesR` 1083, 1399
`\addtocontents` 1769–1771
`\addtocounter` 926, 948, 1020
`\advanceline` 620, 651
`\affixline@num` 991
`\affixline@numR` 1078, 1204
`\affixpstart@numL` 1005, 1311
`\affixpstart@numR` 1311
`\affixside@note` 993, 1007
`\affixside@noteR` 1080, 1541
`\aftercolumnseparator` . 4, 1940, 1981
`\appto` 1547, 1548
`\araw@textfalse` 1821
`\araw@texttrue` 1821
`astanza (environment)` 10, 1663
`\at@begin@pairs` 741, 746, 747
`\at@every@pend` 930, 952
`\at@every@pstart` 861, 908
`\AtBeginDocument` ... 1518, 1754, 1779
`\AtBeginPairs` 4, 746
- B**
- `\ballast@count` 1132, 1137
`\bbl@main@language` 1795, 1796
`\bbl@set@language` 1786, 1787
`\beforecolumnseparator`
 4, 1934, 1937, 1943, 1981
`\beginnumbering` 8, 788, 830
`\beginnumberingR` ... 35, 116, 788, 878
`\bfseries` 812, 814
`\boolfalse` 1442, 1486
`\booltrue` 1445, 1489
`\bypage@Rfalse` 136, 149, 154
`\bypage@Rtrue` 136, 144
`\bypstart@Rfalse` 136, 145, 155
`\bypstart@Rtrue` 136, 150
- C**
- `\c@ballast` 1137
`\c@chapter` 97
`\c@chapterR` 97
`\c@firstlinenumR` 178, 1277
`\c@firstsublinenumR` 182, 1272
`\c@linenumincrementR` 178, 1277
`\c@page` 604, 606, 1430, 1432,
 1474, 1476, 2244, 2250, 2254, 2263
`\c@pstartL` 716, 728, 812
`\c@pstartR` 721, 734, 814
`\c@section` 98
`\c@sectionR` 98
`\c@sublinenumincrementR` .. 182, 1272
`\c@subsection` 99
`\c@subsectionR` 99
`\c@subsubsection` 100
`\c@subsubsectionR` 100
`\ch@ck@l@ckR` 1204
`\ch@cksub@l@ckR` 1204
`\ch@cksub@lockR` 1273
`\chapter` 750, 751, 760
`\chapterinpages` 737, 751, 762
`\chardef` 1660
`\check@goal` 2322, 2343, 2406
`\check@pstarts`
 1809, 1860, 1911, 2093, 2101, 2109
`\checkpageL` 2122, 2150, 2315
`\checkpageR` 2162, 2191, 2315
`\checkpb@columns` ... 1907, 1948, 1952
`\checkpbl` 2152, 2361
`\checkpbr` 2193, 2361
`\checkraw@text`
 1821, 1868, 1904, 2119, 2203
`\checkverseL` 1905, 2151, 2384
`\checkverseR` 1906, 2192, 2384
`\cleardoublepage` 2257
`\clearl@dleftpage` 2161, 2251
`\clearl@drightpage` 2202, 2251
`\cleartoevenpage` 2242, 2266
`\cleartol@devenpage` 2080, 2242
`\closeout` 88, 592, 599
`\columnrulewidth` 4, 1972
`\Columns` 3, 1846
`\columns@position` .. 1892, 1902, 1977
`\columnseparator` 4, 1939, 1972
`\columnspan` 4, 1977
`\correct@footinsX@box` 1423
`\correct@Xfootins@box` 1423
`\correcthangingL` 1016, 1641
`\correcthangingR` 1641
`\count` 1438, 1446, 1482, 1490
`\countLline` 961, 973
`\countRline` 961, 1060
`\critext` 656
`\cs` 1452, 1949, 2438
`\csdingdef` 2596, 2597
`\csgdef` 678, 679, 861,
 862, 908, 909, 930, 931, 952, 953
`\cslet` 690, 695, 717, 722

\csundef 1036, 1877, 1888,
2127, 2148, 2168, 2189, 2441, 2476
\csuse 685, 688, 698, 699, 1033, 1425,
1426, 1436–1441, 1446, 1447,
1449, 1450, 1458, 1460, 1461,
1463, 1469, 1470, 1480–1485,
1490, 1491, 1493, 1494, 1501,
1503, 1504, 1506, 1610, 1621,
1875, 1886, 2126, 2147, 2167,
2188, 2227, 2228, 2440, 2475,
2513, 2514, 2521, 2522, 2536–
2538, 2543–2545, 2571, 2583, 2596

D

\DeclareOption 9–12
\def@tempb 152
\dimdef 2552, 2558, 2559,
2571, 2585, 2586, 2588, 2592, 2593
\dimen 607,
608, 612–614, 618, 1441, 1450,
1485, 1494, 2137–2139, 2178–2180
\dimexpr 1461, 1462, 1504, 1505
\dingdef 2554, 2555
\divide 1209, 1996, 2017, 2033, 2045, 2060
\do@actions 1114
\do@actions@fixedcodeR 1141
\do@actions@nextR 1141
\do@actionsR 1096, 1141
\do@ballast 1115
\do@ballastR 1097, 1132
\do@insidelineLhook . 1010, 1049, 1052
\do@insidelineRhook 1050, 1052
\do@lineL 971, 1871, 2128
\do@lineLhook 977, 1047, 1052
\do@lineR 1057, 1882, 2169
\do@lineRhook 1048, 1052, 1064
\do@lockoff 503
\do@lockoffL 527
\do@lockoffR 503
\do@lockon 468
\do@lockonL 500
\do@lockonR 468
\doinsidelinehook 2022, 2050
\doinsidelineLhook 1047
\doinsidelineRhook 1047
\dolineLhook 1047
\dolineRhook 1047
\dolistloop 710, 1552, 1572, 1579, 2237
\dummy@ref 549
\dump@pstartL@pc 715, 924

\dump@pstartR@pc 715, 946

E

\edfont@info 660, 663, 669, 672
\edlabel 1512
\edtext 656
\eled@sectioningR@out .. 63, 88, 2497
\eled@sections@@ 992, 1024, 1872, 2129
\eled@sectionsR@@ 60, 1079, 1883, 2170
\eledpar@error 19, 21, 24, 27, 29
\eledsection@correcting@skip ...
..... 1044, 1848, 2071, 2496
\eledsectmark 13, 2494
\eledsectnotoc 13, 2492
\empty 73, 76, 267, 275,
658, 667, 726, 732, 838, 886,
1194, 1276, 1284, 1380–1382,
1393, 1404, 2270, 2276, 2283, 2289
\endashchar 1415
\endgraf 919, 941, 1854, 2083
\endline@num 558, 564
\endlock 640, 1669, 1677, 1687
\endnumbering 8, 66, 120, 789
\endnumberingR 38, 66, 103, 115, 128, 789
\endpage@num 557, 564
\endstanzaextra 1689
\endsub 607
\endsubline@num 559, 565
\enlargethispage 1969
environments:
 astanza 10, 1663
 Leftside 7, 769
 pages 5, 737
 pairs 3, 737
 Rightside 7, 786
\expandonce 1591, 1595, 1611, 1622
\extensionchars 55, 109, 125, 133

F

\f@x@l@cksR 1204
\finish@Pages@notes 2222, 2225
\first@linenum@out@Rfalse .. 587, 593
\first@linenum@out@Rtrue 587
\firstlinenum 7, 187
\firstlinenum* 7, 187
\firstsublinenum 7, 187
\firstsublinenum* 7, 187
\fix@page 328, 335
\flag@end 607
\flag@start 607

`\flush@notes` 1921, 2212
`\flush@notesR` 1402, 1922, 2213
`\footnotelang@lua` 1605, 1616
`\footnotelang@poly` 1609, 1620
`\footnoteXmk` 6
`\footnoteXnomk` 6
`\fullstop` . 226, 1412, 1414, 1416, 1418

G

`\get@linelistfile` 263
`\get@nextboxL` 2145, 2419
`\get@nextboxR` 2186, 2419
`\getline@numL` 983, 1112
`\getline@numR` 1070, 1094
`\getlinesfrompagelistL`
 2105, 2205, 2282
`\getlinesfrompagelistR`
 2106, 2206, 2282
`\getlinesfromparlistL` ... 2097, 2269
`\getlinesfromparlistR` ... 2098, 2269
`\gl@p` .. 270, 271, 278, 279, 662, 671,
 694, 727, 733, 1197, 1198, 1386,
 1390, 1405, 2273, 2279, 2286, 2292
`\goalfraction` 5, 2406

H

`\hangingsymbol` 11, 1632, 1638
`\hb@xt@` 990, 999, 1009, 1077,
 1087, 1891, 2134, 2143, 2175, 2184
`\hsize` 856, 903, 1891, 1991–
 1994, 1997, 2011, 2013, 2014,
 2016, 2018, 2029, 2040–2043,
 2046, 2056, 2135, 2143, 2176, 2184

I

`\if@filesw` 1767
`\if@firstcolumn` 1290, 1315, 1336, 1558
`\if@nobreak` 817, 865
`\if@noeled@sec` 61, 87
`\if@nopb` 1954, 1969
`\if@pb` 1953, 1970
`\if@pstarts` 1809, 1861, 2094, 2110
`\if@RTL` 1037
`\ifaraw@text` ... 1821, 1869, 2121, 2204
`\ifautopar` 848, 896
`\ifbool` 1457, 1500
`\ifboolexpr` 1429, 1473
`\ifbypage@` 351
`\ifbypage@R` 136, 341, 1176
`\ifbypstart@` 572, 1912, 2434

`\ifbypstart@R` ... 136, 576, 1916, 2471
`\ifcseempty` 1424, 1468
`\ifcsstring` 1430, 1432, 1474, 1476
`\ifdefstring` 1021, 1022, 1031,
 1874, 1885, 1892, 1902, 2125, 2166
`\ifdim` 608,
 612, 614, 618, 1934, 1940, 2134,
 2175, 2246, 2252, 2261, 2323, 2344
`\ifdimgreater` 2574, 2577
`\ifdimless` 2587
`\iffirst@linenum@out@R` 587, 591
`\ifhbox` 1894, 1899
`\ifinserterhangingsymbol`
 1630, 1644, 2387
`\ifinserterhangingsymbolR`
 1628, 1636, 1654, 2398
`\ifinstanzaL` 767, 767, 1631, 1643, 2385
`\ifinstanzaR` 767, 768, 1637, 1653, 2396
`\ifl@d@dash` 1415
`\ifl@d@elin` 1417, 1418
`\ifl@d@esl` 1418
`\ifl@d@pnum` 1412, 1416
`\ifl@d@ssub` 1414
`\ifl@dpagefull` 2154, 2195, 2315
`\ifl@dpageing` 14, 1642, 1652
`\ifl@dpairing` 14, 70
`\ifl@dsamelang` 1764
`\ifl@dsamepage` 2124, 2164, 2315
`\ifl@dskipnumber` 1267
`\ifl@dusedbabel` 1763
`\iflabelpstart` 858, 905
`\ifledgroupnotesL@` 1116
`\ifledgroupnotesR@` 1098, 1266
`\iflednopbinverse` 2386, 2397
`\ifledplinenum` 1413
`\ifledRcol` . 14, 170, 192, 196, 200,
 204, 248, 265, 329, 338, 363,
 377, 394, 411, 423, 431, 452,
 497, 524, 533, 542, 609, 615,
 621, 628, 636, 641, 645, 650,
 657, 1523, 1590, 1603, 1790, 1802
`\ifluatex`
 823, 871, 1011, 1027, 1604, 1615
`\ifnoteschanged@` 80
`\ifnumberedpar@`
 832, 880, 915, 937, 1588, 1602
`\ifnumbering` 139, 828, 912
`\ifnumberingR` ... 36, 67, 105, 876, 934
`\ifnumberline` 1098, 1116, 1266

- \ifnumberstart 717,
722, 849, 897, 925, 947, 972, 1058
 - \ifnumequal 1545
 - \ifnumgreater 1553, 1573, 1580
 - \ifnumodd 1430, 1432, 1474, 1476
 - \ifodd 1300, 1325,
1346, 1568, 2244, 2250, 2254, 2263
 - \ifparledgroup 2563, 2595
 - \ifprint@last@after@pendL
..... 956, 2146, 2333
 - \ifprint@last@after@pendR
..... 956, 2187, 2354
 - \ifpst@rtedL 32, 836
 - \ifpst@rtedR 32, 884
 - \ifpstartnum 1356, 1361
 - \ifpstartnumR 1311
 - \ifshiftedpstarts 6, 2133, 2174, 2407
 - \ifsidepstartnum 850, 898, 1313, 1334
 - \ifstrempty 860, 907, 929, 951
 - \IfStrEq 981, 1068, 1955,
1963, 2362, 2366, 2374, 2378,
2389, 2390, 2400, 2401, 2513,
2514, 2521, 2522, 2534, 2535, 2542
 - \ifstrequal 2566, 2572
 - \ifsublines@ 224,
317, 362, 395, 402, 433, 442,
454, 461, 473, 507, 563, 565,
1099, 1117, 1183, 1270, 1525, 1529
 - \ifvbox 974, 1061, 1824, 1827, 2420, 2457
 - \ifvoid 2227, 2228
 - \ifwidthliketwocolumns 12
 - \ifwrittenlinesL 2416, 2428
 - \ifwrittenlinesR 2417, 2465
 - \init@series@eledpar 708
 - \initnumbering@sectcmd 740, 754
 - \initnumbering@sectcountR
..... 59, 92, 112, 2165
 - \InputIfFileExists 62
 - \insert@count 539, 1597, 1624
 - \insert@countR 540, 1593, 1613
 - \insert@noterule@ledgroup
..... 2517, 2525, 2533
 - \inserthangingsymbolfalse 987
 - \inserthangingsymbolL ... 1015, 1628
 - \inserthangingsymbolR 1628
 - \inserthangingsymbolRfalse 1074
 - \inserthangingsymbolRtrue 1072
 - \inserthangingsymboltrue 985
 - \insertlines@listR
.... 73, 234, 250, 545, 1382, 1386
 - \inserts@list 837, 1596, 1623
 - \inserts@listR 885, 1377,
1380, 1390, 1404, 1405, 1592, 1612
 - \instanzaLfalse 1930, 2219
 - \instanzaLtrue 778
 - \instanzaRfalse 1931, 2220
 - \instanzaRtrue 801
 - \itemcount@ 1543, 1545,
1550, 1553, 1571, 1573, 1578, 1580
- L**
- \l@d@nums 660, 663, 669, 672
 - \l@d@set 410, 636, 637
 - \l@dbbl@set@language 1764, 1787
 - \l@dbfnote 1587
 - \l@dc@maxchunks 843, 845,
891, 893, 1722, 1732, 1740, 1747
 - \l@dcalc@maxoftwo
..... 2099, 2302, 2442, 2477
 - \l@dcalc@minoftwo .. 2107, 2207, 2302
 - \l@dcalcnun 1204
 - \l@dchecklang 1764
 - \l@dchset@num 284, 287, 410
 - \l@dcsnotetext 1552, 1555
 - \l@dcsnotetext@l 1555, 1572
 - \l@dcsnotetext@r 1555, 1579
 - \l@demptyd@ta 978, 1065
 - \l@dend@stuff 56, 110, 126, 134
 - \l@dgetline@margin 168
 - \l@dgetsidenote@margin 1534
 - \l@dld@ta 1006,
1291, 1303, 1316, 1328, 1337, 1349
 - \l@dleftbox 958,
989, 999, 1893, 2134, 2135, 2143
 - \l@dlinenumR 216
 - \l@dlsn@te 1008
 - \l@dmake@labelsR 1512
 - \l@dminpagelines
.... 2065, 2108, 2208, 2324, 2345
 - \l@dnumpstartsL 717, 842,
843, 845, 847, 861, 862, 930,
931, 1726, 1758, 1812, 1849,
1850, 1926, 2075, 2076, 2217, 2432
 - \l@dnumpstartsR 40, 722, 890,
891, 893, 895, 908, 909, 952,
953, 1726, 1759, 1815, 1849,
1850, 1927, 2075, 2076, 2218, 2469
 - \l@doldbbl@set@language 1786
 - \l@doldselectlanguage 1785, 1789, 1794

- \l@dpagfullfalse 2315, 2364, 2369, 2376, 2381
- \l@dpagfulltrue 2315, 2363, 2368, 2375, 2380
- \l@dpagingfalse 739, 759
- \l@dpagingtrue 753
- \l@dpairingfalse 743, 758
- \l@dpairingtrue 738, 752
- \l@dprintingcolumnfalse 1928
- \l@dprintingcolumnstrue 1847
- \l@dprintingpagesfalse 2221
- \l@dprintingpagetrue 2070
- \l@dpscl 972,
974, 979, 992, 1023, 1033, 1036,
1728, 1760, 1812, 1824, 1858,
1864, 1872, 1875, 1877, 1924,
2087, 2095, 2100, 2103, 2111,
2126, 2127, 2129, 2147, 2148,
2215, 2420, 2422, 2425, 2432,
2440–2442, 2444, 2446, 2512, 2567
- \l@dpscR 1058, 1061, 1066,
1079, 1729, 1761, 1815, 1827,
1859, 1865, 1883, 1886, 1888,
1925, 2088, 2096, 2104, 2112,
2167, 2168, 2170, 2188, 2189,
2216, 2457, 2459, 2462, 2469,
2475–2477, 2479, 2481, 2520, 2569
- \l@drd@ta 1016,
1293, 1301, 1318, 1326, 1339, 1347
- \l@drightbox 958,
1076, 1087, 1898, 2175, 2176, 2184
- \l@drsn@te 1017
- \l@dsamepagefalse 2315, 2363, 2368, 2375, 2380
- \l@dsamepagetrue 2315, 2364, 2369, 2376, 2381
- \l@dsetupmaxlinecounts .. 1739, 1756
- \l@dsetuprawboxes 1731, 1755
- \l@dskipnumberfalse 1268
- \l@dskipnumbertrue 1164
- \l@dunhbox@line 1016, 1039, 1042
- \l@dusedbabelfalse 1763, 1782
- \l@dusedbabeltrue 1763, 1784
- \l@duselanguage 1774, 1870, 1881, 2123, 2163
- \l@dzeroxlinecounts ... 1739, 1757
- \l@dzeroopenalties 918, 940, 1853, 2082
- \l@luatexbodydir@L 826, 1030
- \l@luatexbodydir@R 874
- \l@luatexpardir@L 825, 1029
- \l@luatexpardir@R 873
- \l@luatextextdir@L 824, 1012, 1028, 1031
- \l@luatextextdir@R 872
- \l@prev@nopbR ... 47, 2499, 2506, 2507
- \l@prev@pbR 46, 2498, 2504, 2505
- \l@pscL 1728
- \l@pscR 1728
- \labelref@list 1530
- \labelref@listR 1510, 1526
- \language name .. 1765, 1766, 1768–1771
- \last@page@num 349, 355
- \last@page@numR 335
- \lastbox 982, 1069
- \lastskip 607, 613
- \Lcolwidth 4,
5, 14, 755, 856, 990, 999, 1878,
1992, 2013, 2030, 2041, 2057, 2078
- \led@err@BadLeftRightPstarts ... 23, 1850, 2076
- \led@err@LeftOnRightPage ... 26, 2255
- \led@err@LineationInNumbered ... 140
- \led@err@ManyLeftnotes 1573
- \led@err@ManyRightnotes 1580
- \led@err@ManySidenotes 1553
- \led@err@NumberingNotStarted ... 84
- \led@err@numberingShouldHaveStarted 114
- \led@err@NumberingStarted 37
- \led@err@PendNoPstart 916, 938
- \led@err@PendNotNumbered ... 913, 935
- \led@err@PstartInPstart ... 833, 881
- \led@err@PstartNotNumbered . 829, 877
- \led@err@RightOnLeftPage ... 26, 2264
- \led@err@TooManyPstarts 20, 844, 892
- \led@mess@NotesChanged 81
- \led@mess@SectionContinued 108, 124, 132
- \led@nopbnumR 2503, 2504
- \led@nopbR 2502, 2504
- \led@pb@setting 1955, 1963, 2362, 2366,
2374, 2378, 2389, 2390, 2400, 2401
- \led@pbnumR 2501, 2504
- \led@pbR 2500, 2504
- \led@warn@BadAction 1166
- \led@warn@BadAdvancelineLine 380, 386
- \led@warn@BadAdvancelineSubline . 366, 372
- \led@warn@BadLineation 157

- \led@warn@BadSetline 626
 - \led@warn@BadSetlinenum 634
 - \led@warn@DuplicateLabel 1514
 - \ledgroupnotesL@false 2528
 - \ledgroupnotesL@true 2516
 - \ledgroupnotesR@false 2531
 - \ledgroupnotesR@true 2524
 - \ledllfill 1009
 - \lednopb 797
 - \lednopbnum 2389, 2500
 - \lednopbnumR 2400, 2500
 - \lednopbR 797, 2502
 - \ledpb 796
 - \ledpbnum 2390
 - \ledpbnumR 2401, 2500
 - \ledpbR 796, 2500
 - \ledRcol@false 1090
 - \ledRcol@true 1059
 - \ledRcolfalse 770, 803
 - \ledRcoltrue 787
 - \ledrlfill 1016, 2001, 2023
 - \ledsavedprintlines 9, 1410
 - \ledsectnomark 1022
 - \ledsectnotoc 1021, 2125, 2166
 - \ledstrutL 2135, 2143, 2239
 - \ledstrutR 2176, 2184, 2239
 - \ledthegoal 2323, 2344, 2406
 - \leftlinenumR 216, 1291, 1303
 - \leftpstartnumL 1311
 - \leftpstartnumR 1311
 - Leftside (environment) 7, 769
 - \Leftsidehook 776, 781
 - \Leftsidehookend 780, 781
 - \letcs 687, 693, 696, 972, 1058
 - \line@list 667, 671
 - \line@list@stuff 125
 - \line@list@stuffR .. 55, 109, 133, 589
 - \line@listR .. 76, 234, 249, 565, 658, 662
 - \line@margin 173, 1321
 - \line@marginR 166, 1296, 1342
 - \line@num .. 352, 384, 385, 387, 405,
416, 417, 445, 572, 1123, 1528, 2435
 - \line@numR 48,
223, 230, 289, 323, 342, 378,
379, 381, 398, 412, 413, 436,
558, 562, 576, 1105, 1177, 1186,
1275, 1277, 1279, 1280, 1524, 2472
 - \lineation 162, 163, 798
 - \lineation* 8, 162
 - \lineationR 8, 138, 164, 798
 - \linenum@out ... 604, 610, 616, 622,
629, 637, 642, 646, 1835, 1913, 2296
 - \linenum@outR 586,
592, 594, 599, 600, 606, 609,
615, 621, 628, 636, 641, 645,
650, 1840, 1917, 2299, 2500–2503
 - \linenumberlist 1276, 1280
 - \linenumincrement 7, 187
 - \linenumincrement* 7, 187
 - \linenummargin 166
 - \linenumr@p ... 1413, 1417, 1524, 1528
 - \linenumrepR 213, 223
 - \linenumsep
. 218, 220, 1358, 1361, 1370, 1373
 - \linesinpar@listL
..... 241, 259, 573, 2270, 2273
 - \linesinpar@listR
..... 241, 253, 577, 2276, 2279
 - \lineskip 1462, 1505
 - \linesonpage@listL 260, 581, 2283, 2286
 - \linesonpage@listR 254, 584, 2289, 2292
 - \list@clear
. 249–256, 259, 260, 262, 837, 885
 - \list@clearing@reg 258
 - \list@create 234–239,
241–243, 682, 713, 714, 1377, 1510
 - \list@pstartL@pc .. 713, 716, 726, 727
 - \list@pstartR@pc .. 713, 721, 732, 733
 - \listbreak 2230, 2234
 - \listxadd 357, 2504–2507
 - \lock@disp 1236, 1240, 1245
 - \lock@off 494, 495, 503, 645, 646
 - \lock@on 641, 642
 - \luatexbodydir 826, 874, 1030
 - \luatexpardir 825, 873, 1029
 - \luatextextdir .. 824, 872, 1012, 1028
- M**
- \managestanza@modulo 1698
 - \maxchunks 3, 1722
 - \maxlinesinpar@list 241, 262
 - \memorydump 8, 775, 792
 - \memorydumpL 119, 775
 - \memorydumpR 119, 792
 - \message 54
 - \multiply 1210, 1995, 2044
- N**
- \n@num 531, 650
 - \n@num@reg 537

- \namebox 974, 979, 1061,
1066, 1706, 1824, 1827, 2420, 2457
 - \NeedsTeXFormat 2
 - \new@line 1039, 1042
 - \new@lineL 603, 1014
 - \new@lineR 605
 - \newbool 680, 681
 - \newbox 808, 958, 959, 1707
 - \newcommandx .. 816, 864, 911, 933, 1686
 - \newcounter 92–
95, 178, 180, 182, 184, 811, 813
 - \newhookcommand@series 704, 706
 - \newif 6, 33, 136,
137, 587, 767, 768, 956, 957,
1365, 1628, 1763, 1809, 1821,
1953, 1954, 2315, 2317, 2416, 2417
 - \newlength 1981, 1984
 - \newmarks 2508–2510
 - \newnamebox 1706, 1734, 1735
 - \newnamecount 1717, 1742
 - \newsavebox 1844, 1845
 - \newseries@eledpar 677, 709
 - \newwrite 586, 2497
 - \next@absline
.... 1956, 1958, 1960, 2367–2369
 - \next@abslineR
.... 1957, 1959, 1961, 2379–2381
 - \next@action 279
 - \next@actionline 276, 278
 - \next@actionlineR
. 268, 270, 1135, 1173, 1195, 1197
 - \next@actionR 271, 1136,
1174, 1175, 1180, 1181, 1189, 1198
 - \next@insert 838
 - \next@insertR
886, 1381, 1384, 1386, 1389, 1393
 - \next@page@num 356, 428
 - \next@page@numR .. 52, 292, 294, 346, 425
 - \noindent 862, 909, 931, 953
 - \normal@page@break 357
 - \normal@page@breakR 45
 - \normal@pars 69, 841, 889
 - \normalbfnoteX 1601
 - \notefontsetup 2554
 - \noteschanged@true
..... 74, 77, 659, 668, 1383
 - \notesXwidthliketwocolumns 4
 - \num@lines 919, 1854, 2083
 - \num@linesR 807, 941, 1855, 2084
 - \numberedpar@true 857, 904
 - \numberingRfalse 68
 - \numberingRtrue 42, 103, 129
 - \numberingtrue 121
 - \numberpstartfalse 9
 - \numberpstarttrue 9
 - \numdef 688, 1023, 1571, 1578,
1956, 1957, 2367, 2379, 2567, 2569
 - \numgdef 1543, 1550, 2388, 2399
 - \numlabfont 223
 - \numpagelinesL 2065,
2132, 2155, 2159, 2324, 2390, 2587
 - \numpagelinesR
2065, 2173, 2196, 2200, 2345, 2401
- O**
- \old@footnote 696, 700
 - \old@otherlanguage 1799
 - \old@startstanza .. 777, 778, 800, 801
 - \oldchapter 750, 760
 - \one@line
. 979, 982, 1009, 1016, 1039, 1042
 - \one@lineR 807, 1066, 1069
 - \onlysideX 6
 - \onlyXside 6
 - \openout 63, 594, 600
 - \otherlanguage 1799, 1800
- P**
- \p@pstartL 859
 - \p@pstartR 906
 - \PackageError 19
 - \page@action 293, 422, 551
 - \page@num 274, 354, 1323
 - \page@numR 245, 266, 344,
557, 562, 1175, 1298, 1344, 1566
 - \pagebreak 1970
 - \pagegoal 2414
 - \Pages 5, 2069
 - pages (environment) 5, 737
 - \pagetotal 2137, 2139, 2178,
2180, 2246, 2252, 2261, 2323, 2344
 - pairs (environment) 3, 737
 - \paperwidth 1038, 1041
 - \par@line 920, 1856, 2085
 - \par@lineR 807, 942, 1857, 2086
 - \parbox 1878, 1889
 - \parledgroup@ .. 981, 1068, 2508, 2534
 - \parledgroup@beforenotes@save ..
..... 928, 950, 2594
 - \parledgroup@beforenotesL 2592

- \parledgroup@beforenotesR [2592](#)
 - \parledgroup@correction@notespacing [2142](#), [2183](#), [2581](#)
 - \parledgroup@correction@notespacing@final [2450](#), [2485](#), [2562](#)
 - \parledgroup@correction@notespacing@initial [2160](#), [2201](#), [2557](#), [2565](#)
 - \parledgroup@notes@endL [2423](#), [2426](#), [2449](#), [2527](#)
 - \parledgroup@notes@endR [2460](#), [2463](#), [2484](#), [2530](#)
 - \parledgroup@notes@startL [981](#), [2511](#), [2527](#)
 - \parledgroup@notes@startR [1068](#), [2511](#), [2527](#)
 - \parledgroup@notespacing@correction [2552](#), [2584](#)–[2586](#)
 - \parledgroup@notespacing@correction@accumulated [2558](#), [2564](#), [2585](#)
 - \parledgroup@notespacing@correction@modulo [2559](#), [2586](#)–[2588](#)
 - \parledgroup@notespacing@set@correction [2072](#), [2552](#)
 - \parledgroup@series [2509](#), [2513](#), [2514](#), [2521](#), [2522](#), [2537](#), [2538](#), [2544](#), [2545](#)
 - \parledgroup@type [2510](#), [2513](#), [2514](#), [2521](#), [2522](#), [2535](#), [2542](#)
 - \parledgroupnotespacing [2551](#), [2554](#)
 - \parledgroupseries@ [2508](#)
 - \parledgrouptrue [11](#)
 - \parledgrouptype@ [2508](#)
 - \pausenumbering [790](#)
 - \pausenumberingR [102](#), [790](#)
 - \pend [7](#), [774](#), [795](#), [834](#), [1688](#)
 - \pendL [774](#), [911](#)
 - \pendR [795](#), [882](#), [933](#)
 - \prev@abslineverse [2388](#)–[2390](#), [2399](#)–[2401](#)
 - \prev@nopbR [2498](#)
 - \prev@pbR [2498](#)
 - \prevgraf [919](#), [941](#), [1854](#), [1855](#), [2083](#), [2084](#)
 - \print@columnseparator [1897](#), [1933](#)
 - \print@eledsectionL [1019](#), [1878](#), [2130](#)
 - \print@eledsectionR [1094](#), [1889](#), [2171](#)
 - \print@last@after@pendLfalse [2336](#)
 - \print@last@after@pendLtrue [2452](#)
 - \print@last@after@pendRfalse [2357](#)
 - \print@last@after@pendRtrue [2487](#)
 - \print@lineL [994](#), [1004](#)
 - \print@lineR [1081](#), [1094](#)
 - \print@notesX [2092](#)
 - \print@notesX@forpages [1423](#), [2092](#)
 - \print@Xnotes [2091](#)
 - \print@Xnotes@forpages [1423](#), [2091](#)
 - \printlines [1421](#)
 - \printlinesR [9](#), [1410](#)
 - \ProcessOptions [13](#)
 - \protected@csxdef [698](#)
 - \protected@edef [858](#), [905](#)
 - \protected@write [1768](#)
 - \protected@xdef [1610](#), [1621](#)
 - \ProvidesPackage [3](#)
 - \pst@rtedLfalse [32](#)
 - \pst@rtedLtrue [122](#), [839](#)
 - \pst@rtedRfalse [41](#), [71](#)
 - \pst@rtedRtrue [106](#), [130](#), [887](#)
 - \pstart [7](#), [21](#), [25](#), [772](#), [794](#), [1682](#)
 - \pstartL [772](#), [810](#)
 - \pstartnumfalse [1358](#), [1363](#)
 - \pstartnumRfalse [1370](#), [1375](#)
 - \pstartnumRtrue [1366](#), [1863](#), [2482](#)
 - \pstartnumtrue [1862](#), [2447](#)
 - \pstartR [794](#), [810](#)
- R**
- \Rcolwidth [4](#), [5](#), [14](#), [756](#), [903](#), [1077](#), [1087](#), [1889](#), [1993](#), [2014](#), [2031](#), [2042](#), [2058](#), [2079](#)
 - \read@linelist [247](#), [590](#)
 - \rem@inder [1280](#), [1282](#)–[1284](#)
 - \RequirePackage [5](#)
 - \resetprevline@ [1914](#), [1918](#), [2436](#), [2473](#)
 - \restore@pstartL@pc [725](#), [1866](#), [2113](#), [2448](#)
 - \restore@pstartR@pc [725](#), [1867](#), [2114](#), [2483](#)
 - \resumenumbering [791](#)
 - \resumenumberingR [102](#), [791](#)
 - \rightlinenumR [216](#), [1293](#), [1301](#)
 - \rightpstartnumL [1311](#)
 - \rightpstartnumR [1311](#)
 - Rightside (environment) [7](#), [786](#)
 - \Rightsidehook [781](#), [799](#)
 - \Rightsidehookend [781](#), [804](#)
 - \rlap [1293](#), [1301](#), [1318](#), [1326](#), [1339](#), [1347](#)
 - \Rlineflag [9](#), [211](#), [223](#), [1413](#), [1417](#), [1516](#)
 - \rule [1973](#)

S

- \savebox 1878, 1889
- \sc@n@list 1281, 1283
- \secdef 765
- \section@num 123–125
- \section@numR 30,
43, 54, 55, 62, 63, 107–109, 131–133
- \select@language 1766, 1768–1771, 1805
- \selectlanguage 1774
- \set@line 656
- \set@line@action
..... 286, 391, 400, 407, 430, 553
- \setl@dlp@rbox 1559, 1574, 1576
- \setl@drp@rbox 1561, 1569, 1581
- \setline 624
- \setlinenum 632
- \setnamebox 847, 895, 1706
- \setnotepositionliketwocolumns@C
..... 1987
- \setnotepositionliketwocolumns@L
..... 1987
- \setnotepositionliketwocolumns@R
..... 1987
- \setnotespositionliketwocolumns@C
..... 2026
- \setnotespositionliketwocolumns@L
..... 2004
- \setnotespositionliketwocolumns@R
..... 2053
- \setpositionliketwocolumns@C ...
..... 1987, 2021
- \setpositionliketwocolumns@L ...
..... 1987, 2000
- \setpositionliketwocolumns@R ...
..... 1987, 2049
- \setprintlines 1411
- \setwidthliketwocolumns@C 1987, 2008
- \setwidthliketwocolumns@L 1987, 1987
- \setwidthliketwocolumns@R 1987, 2038
- \shiftedpstartsfalse 8
- \shiftedpstartstrue 7, 9, 10
- \shiftedversesfalse 8
- \shiftedversestrue 7
- \sidenote@margin 1536, 1539
- \sidenote@marginR 1533, 1564
- \sidenotecontent@
..... 1542, 1547, 1548, 1559, 1561,
1569, 1570, 1574, 1576, 1577, 1581
- \sidenotemargin 1533
- \sidenotemargin* 1533
- \sidenotesep 1548
- \skip 1439, 1447, 1483, 1491, 2537, 2544
- \skip@lockoff 495, 503
- \skipnumbering 10, 649
- \skipnumbering@reg 653
- \smash 1973
- \splitbotmarks 2534,
2535, 2537, 2538, 2542, 2544, 2545
- \splitfirstmarks
981, 1068, 2513, 2514, 2521, 2522
- \splittopskip .. 976, 1063, 1459, 1502
- \stanza@count 1666, 1679, 1693
- \stanza@hang 1668, 1701
- \stanza@modulo 1666, 1696
- \stanzaindentbase
..... 1646, 1656, 1694, 1697
- \startlock 640
- \startstanzahook 1664
- \startsub 607
- \sub@action 302, 451, 552
- \sub@change 53, 296, 297, 303
- \sub@lock 1118
- \sub@lockR 50, 311, 313, 315,
318, 469, 475, 476, 478, 479,
509, 510, 512, 1100, 1156, 1158,
1159, 1161, 1217, 1257, 1259, 1261
- \sub@off 615, 616
- \sub@on 609, 610
- \subline@num 225, 352, 370,
371, 373, 403, 443, 1119, 1124, 1529
- \subline@numR 226,
230, 319, 323, 342, 364, 365,
367, 396, 434, 559, 563, 1101,
1106, 1177, 1184, 1271, 1272, 1525
- \sublinenumincrement 7, 187
- \sublinenumincrement* 7, 187
- \sublinenumr@p . 1414, 1418, 1525, 1529
- \sublinenumrepR 213, 226
- \sublines@false 51, 300, 1146
- \sublines@true 298, 1144
- \sublock@disp 1219, 1223, 1228
- \sw@list@inedtextR 239, 256
- \sw@listR 238, 255
- \symplinenum 1413
- \sza@penalty 1674, 1678

T

- \temp 1990, 1991, 1994–1997,
2010, 2011, 2016–2018, 2027,

2029, 2032–2035, 2039, 2040, 2043–2046, 2054, 2056, 2059–2062	<code>\vbadness</code> 975, 1062
<code>\temp@</code> 1023, 1024	<code>\vbfnoteX</code> 1611, 1622
<code>\temp@spacing</code> 2554, 2555	<code>\vbox</code> . . 847, 895, 1460, 1503, 1878, 1889
<code>\tempa</code> 2028, 2030–2032, 2055, 2057–2059	<code>\vl@dbfnote</code> 1591, 1595
<code>\textwidth</code> 15, 17, 755, 756, 2078, 2079	<code>\vsplit</code> 979, 1066, 1461, 1504
<code>\theledlanguageL</code> . . . 1774, 1870, 2123	W
<code>\theledlanguageR</code> . . . 1774, 1881, 2163	<code>\wd</code> 1009
<code>\thepage</code> 604, 606	<code>\widthliketwocolumns</code> 4
<code>\thepstart</code> 773, 793	<code>\widthliketwocolumnstrue</code> 12
<code>\thepstartL</code>	<code>\WithSuffix</code> 162, 207–210, 1533
. 9, 773, 812, 852, 859, 1357, 1362	<code>\writtenlinesLfalse</code> 2089, 2433
<code>\thepstartR</code>	<code>\writtenlinesLtrue</code> 2430
. 9, 793, 814, 899, 906, 1369, 1374	<code>\writtenlinesRfalse</code> 2090, 2470
<code>\thisfootnote</code> . . 1610, 1611, 1621, 1622	<code>\writtenlinesRtrue</code> 2467
<code>\thr@@</code> . . 478, 487, 510, 517, 1151, 1159	X
<code>\togglefalse</code> 686	<code>\xifinlist</code> 992,
<code>\toggletrue</code> 684	1024, 1079, 1872, 1883, 2129, 2170
<code>\topskip</code> 2246	<code>\xifinlistcs</code> 1958–
U	1961, 1964–1967, 2363, 2364,
<code>\unhbox</code> 1711,	2368, 2369, 2375, 2376, 2380, 2381
1893, 1898, 2135, 2143, 2176, 2184	<code>\Xnoteswidthliketwocolumns</code> 4
<code>\unhnamebox</code> 1706	<code>\xpg@main@language</code> 1806, 1807
<code>\unvbox</code> . . . 982, 1069, 1463, 1506, 1713	<code>\xright@appenditem</code> 424,
<code>\unvnamebox</code> 1706	425, 427, 428, 432, 439, 441,
<code>\usebox</code> 1895, 1900	448, 453, 455, 457, 460, 462,
<code>\usenamecount</code> 1667, 1673, 1717, 1749,	464, 472, 474, 483, 506, 508,
2100, 2422, 2425, 2442, 2444,	515, 534, 535, 545, 561, 573,
2459, 2462, 2477, 2479, 2512, 2520	577, 581, 584, 689, 716, 721,
V	1524, 1528, 1591, 1595, 1611, 1622
<code>\value</code> 1692	<code>\xspace</code> 701

Change History

v0.1.	v0.3.
General: First public release 1	General: Added <code>\do@lineLhook</code>
v0.2.	and <code>\do@lineRhook</code> 46
General: Added section of babel re-	Reorganize for ledarab 1
lated code 66	<code>\affixline@numR:</code> Changed
Fix babel problems 1	<code>\affixline@numR</code> to match new
<code>\Columns:</code> Added <code>\l@dchecklang</code>	<code>eledmac</code> 49
and <code>\l@duselanguage</code> to	<code>\do@actions@nextR:</code> Used
<code>\Columns</code> 70	<code>\do@actions@fixedcode</code> in
<code>\Pages:</code> Added <code>\l@duselanguage</code>	<code>\do@actionsR</code> 48
to <code>\Pages</code> 77	<code>\do@lineL:</code> Added <code>\do@lineLhook</code>

to <code>\do@lineL</code>	44	v0.3.c.	
Simplified <code>\do@lineL</code> by using macros for some common code	44	General: Compatibilty with Poly- glossia	1
<code>\do@lineR</code> : Changed <code>\do@lineR</code> similarly to <code>\do@lineL</code>	46	v0.4.	
Leftside: Added hooks into Left- side environment	38	General: No more ledparpatch. All patches are now in the main file.	1
<code>\flag@end</code> : Removed extraneous spaces from <code>\flag@end</code>	32	v0.5.	
<code>\ifledRcol</code> : Moved <code>\ifl@dpairing</code> to <code>eledmac</code>	15	General: Corrections about <code>\section</code> and other titles in numbered sections	1
<code>\ifpst@rtedR</code> : Moved <code>\ifpst@rtedL</code> to <code>eledmac</code>	16	v0.6.	
<code>\l@dlinenumR</code> : Simplified <code>\leftlinenumR</code> and <code>\rightlinenumR</code> by introducing <code>\l@dlinenumR</code>	21	General: Be able to use <code>\chapter</code> in parallel pages.	1
<code>\l@dnumpstartsR</code> : Moved <code>\l@dnumpstartsL</code> to <code>eledmac</code> .	65	v0.7.	
<code>\ledsavedprintlines</code> : Simpli- fied <code>\printlinesR</code> by using <code>\setprintlines</code>	55	General: Option ‘shiftedverses’ which make there is no blank between two parallel verses with inequal length.	1
<code>\ledstrutR</code> : Added <code>\ledstrutL</code> and <code>\ledstrutR</code>	80	v0.8.	
<code>\normalbfnoteX</code> : Removed extraneous spaces from <code>\normalbfnoteX</code>	62	General: Possibility to have a sym- bol on each hanging of verses, like in the french typogra- phy. Redefine the commande <code>\hangingsymbol</code> to define the character.	1
<code>\Pages</code> : Added <code>\ledstrutL</code> to <code>\Pages</code>	77	v0.9.	
Added <code>\ledstrutR</code> to <code>\Pages</code> .	78	General: Possibility to number <code>\pstart</code>	9
<code>\Rightsidehookend</code> : Added <code>\Leftsidehook</code> , <code>\Leftsidehookend</code> , <code>\Rightsidehook</code> and <code>\Rightsidehookend</code>	38	Possibilty to number the <code>pstart</code> with the commands <code>\numberpstarttrue</code>	1
<code>\sublinenumrepR</code> : Added <code>\linenumrepR</code> and <code>\sublinenumrepR</code>	21	<code>\ifledRcol</code> : Moved <code>\iflledRcol</code> and <code>\ifnumberingR</code> to <code>eledmac</code>	15
v0.3.a.		v0.9.1.	
General: Minor <code>\linenummargin</code> fix	1	General: The numbering of the <code>pstarts</code> restarts on each <code>\beginnumbering</code>	1
<code>\line@marginR</code> : Don’t just set <code>\line@marginR</code> in <code>\linenummargin</code>	19	v0.9.2.	
v0.3.b.		General: Debug : with <code>\Columns</code> , the hanging indentation now runs on the left columns and the hanging symbol is shown only when <code>\stanza</code> is used.	1
General: Improved parallel page balancing	1	v0.9.3.	
<code>\Pages</code> : Added <code>\l@dminpagelines</code> calculation for succeeding page pairs	79	General: <code>\thepstartL</code> and <code>\thepstartR</code> use now <code>\bfseries</code> and not <code>\bf</code> , which is deprecated and makes con- flicts with memoir class.	1

v0.10.		v1.1.1.	
General: <code>\edlabel</code> commands on the right side are now correctly indicated.	1	<code>\pstartR</code> : Correct <code>\pstartR</code> bug introduced by 1.1.	40
<code>\edlabel</code> commands which start a paragraph are now put in the right place.	1	v1.1.2.	
		<code>\affixside@noteR</code> : Remove spurious space between line number and line content	60
v0.11.		v1.2.	
General: Change <code>\do@lineL</code> and <code>\do@lineR</code> to allow line numbering by <code>pstart</code> (like in <code>eledmac 0.15</code>).	44	General: Support for <code>\led<section></code> commands in parallel texts. . . .	1
Lineation can be by <code>pstart</code> (like in <code>eledmac 0.15</code>).	18	v1.2.1.	
New management of hangingsymbol insertion, preventing undesirable insertions.	62	<code>\initnumbering@sectcountR</code> : For the right section, the counter is defined only once.	17
Prevent shift of column separator when a verse is hanged	63	v1.3.	
<code>\affixline@numR</code> : Changed <code>\affixline@numR</code> to allow to disable line numbering (like in <code>eledmac 0.15</code>).	49	<code>\edtext</code> : Manage RTL language. . . .	34
<code>\Columns</code> : Line numbering by <code>pstart</code>	71	v1.3.1.	
<code>\get@nextboxR</code> : Change <code>\get@nextboxL</code> and <code>\get@nextboxR</code> to allow to disable line numbering (like in <code>eledmac 0.15</code>).	85	<code>\l@dbfnote</code> : Compatibility of standard footnotes with <code>eledmac</code> when theses footnotes contain any commands.	61
<code>Pstart</code> number can be printed in side	86	v1.3.2.	
		General: Debug with some classes. . .	1
v0.12.		v1.3.3.	
General: New new management of hangingsymbol insertion, preventing undesirable insertions. . . .	62	General: Debugging the left notes of the right column.	60
v1.0.		<code>\l@dbfnote</code> : Spurious space with footnote in right column.	61
General: Compatibility with <code>eledmac</code> . Change name to <code>eledpar</code> . . .	1	v1.3.4.	
Debug in lineation by <code>pstart</code> . . .	18	General: Allow use of commands in sidenotes, as introduced by <code>eledmac 1.0</code>	60
v1.0.1.		v1.3.5.	
General: Correction on <code>\numberonlyfirstinline</code> with lineation by <code>pstart</code> or by page.	1	<code>\normalbfnoteX</code> : Allows one to redefine <code>\thefootnoteX</code> with <code>alph</code> when some packages are loaded.	62
v1.1.		v1.4.	
General: <code>Shiftedverses</code> becomes <code>shiftedpstarts</code>	1	General: Added <code>\do@insidelineLhook</code> and <code>\do@insidelineRhook</code> . . .	46
<code>\pstartR</code> : Add <code>\labelpstarttrue</code> (from <code>eledmac</code>).	40	v1.4.1.	
		<code>\normalbfnoteX</code> : Fix bug with normal familiar footnotes when mixing RTL and LTR text. . . .	62
		<code>astanza</code> : Enable the use of <code>stanza</code> environment.	63
		v1.4.3.	
		General: Corrects a false hanging verse when a verse is exactly the	

length of a line.	1	New sectioning commands, as in	
<code>\inserthangingsymbolR</code> : Hang		eledmac.	13
verse is now not automatically		<code>\Columns</code> : Modify <code>\Columns</code> to en-	
flush right.	62	able to add section's title. . . .	69
<code>\pendL</code> : Spurious spaces in <code>\pendL</code> .	42	Suppress <code>\l@dchecklang</code> from	
<code>\pendR</code> : Spurious spaces in		<code>\Columns</code>	70
<code>\pstartR</code>	43	<code>\l@dchecklang</code> : Suppress	
<code>\pstartR</code> : Spurious spaces in		<code>\l@dchecklang</code> which didn't	
<code>\pstartL</code> and <code>\pstartR</code>	40	work and was not logical, be-	
v1.5.0.		cause both columns could have	
General: Add, as in eledmac, fea-		the same language but not the	
tures to manage page breaks. . .	1	main language of the docu-	
<code>\sublinenumincrement*</code> : Add		ment.	67
starred version of <code>\firstlinenum</code> ,		<code>\Pages</code> : Modify <code>\Pages</code> to enable to	
<code>\linenumincrement</code> , <code>\firstsublinenum</code> ,		add section's title.	75
<code>\sublinenumincrement</code> to		<code>\pendL</code> : As in eledmac, <code>\pendL</code> can	
change both Left and Right-		have an optional argument. . .	42
side.	20	<code>\pendR</code> : As in eledmac, <code>\pendR</code> can	
v1.6.0.		have an optional argument. . .	43
General: Add tool and documenta-		<code>\print@columnseparator</code> : Move	
tion for parallel ledgroups . . .	12	some code of <code>\Columns</code> to	
v1.7.0.		<code>\print@columnseparator</code> . . .	72
General: Add, as in eledmac, fea-		<code>\pstartR</code> : As in eledmac, <code>\pendL</code>	
tures to make crossrefs with		and <code>\pendR</code> can have an op-	
<code>pstart</code> numbers.	1	tional argument.	40
v1.8.0.		<code>\sidenotemargin*</code> : <code>\sidenotemargin</code>	
General: <code>\beginnumbering</code> is de-		is now directly defined in eled-	
fined only on eledmac, not on		mac to be able to manage eled-	
eledpar.	16	par.	60
<code>\l@dlsnote</code> , <code>\l@drsnote</code> and		Add <code>\sidenotemargin*</code>	60
<code>\l@dcnote</code> defined only one		<code>\theledlanguageR</code> : Correct	
time, in eledmac.	60	left/right language setting with	
Add <code>\beforecolumnseparator</code>		polyglossia.	68
and <code>\aftercolumnseparator</code> . . .	4	v1.8.1.	
Add <code>\columnspan</code>	4	<code>\do@lineL</code> : Fix a bug with critical	
Add, as in eledmac, new system		notes at the beginning of a page,	
of sectioning commands.	1	(maybe added by v1.8.0) (?). .	44
Add, as in eledmac, option to in-		<code>\do@lineR</code> : Fix a bug with critical	
sert something after <code>\pends</code> /		notes at the beginning of a page,	
verses.	1	added by v1.8.0 (?).	46
Add, as in eledmac, option		v1.8.2.	
to insert something between		General: Debug <code>\eledxxx</code> with	
<code>\pstarts</code> / verse.	1	some paper sizes	1
Change <code>\do@lineR</code> and		Debug left and side note (bugs	
<code>\do@lineR</code> to allow new sec-		added by 1.8.0)	1
tioning commands.	44	<code>\eledpar@error</code> : Errors specific to	
Compatibility with musixtex. . .	1	eledpar send to eledpar hand-	
Debug eledmac section-		book	15
ing command after using		<code>\flag@end</code> : <code>\flag@start</code> and	
<code>\resumenumbering</code>	1	<code>\flag@end</code> are now defined only	

one time for eledmac and eled-	vertical mode to solve spurious
par	space before minipage.
\lineation*: Add \lineation*	32
v1.8.3.	v1.11.0.
General: Add \noeledxxx, as in	General: Compatibility of standard
eledmac	footnotes with some biblatex
\doinsidelineRhook: Added	styles.
\dolineLhook, \dolineRhook,	\edtext: \critext and \edtext are
\doinsidelineLhook and	now defined only in eledmac.
\doinsidelineRhook	34
\Pages: Debug blank pages when	v1.12.0.
using optional argument in the	General: Compatibility with
last \pend.	Lua \LaTeX RTL languages.
\resumenumberingR: Debug	\Columns: Add \l@dprintingcolumnstrue
\resumenumberingR
17	69
v1.9.0.	\edlabel: \edlabel and \edindex
General: Add \AtBeginPairs	works now with hyperref when
macro.	using eledpar.
4	59
Compatibility with \Xnoteswidthliketwocolumns	\edlabel is now defined only one
and \notesXwidthliketwocolumns	time for both eledmac and eled-
.	par
1	59
\ifwidthliketwocolumns: Added	\Pages: Add \l@dprintingpagetrue
widthliketwocolumns option
14	75
\theledlanguageR: Debug	\print@eledsectionL: Compati-
left/right language switching	bility with Lua \LaTeX RTL lan-
with polyglossia. Don't write in	guages.
.aux file when setting left/right	45
lines.	\print@eledsectionR: Compati-
68	bility with Lua \LaTeX RTL lan-
v1.9.1.	guages.
\ifledRcol: Moved \ifl@dpadding	47
to eledmac	\print@lineL: Compatibility with
15	Lua \LaTeX RTL languages.
v1.10.0.	45
General: Compatibility with	v1.12.1.
\AtEveryPstart and \AtEveryPend	\print@eledsectionL: Fixes
.	bug with Lua \LaTeX RTL
1	\eledsection.
Restore critical notes in	45
\eledsection in parallel	v1.13.0.
columns (this bug was added	General: Fix bug in shiftedpstarts
in 1.8.2).	when size difference between
1	pstarts is very important.
\Pages: Debug wrong pages split-	1
ting when no optional argument	With parallel pages, long notes
is used in last \pend (bug was	can now flow from the Left
added in v.1.8.3).	to the right side and from the
75	Right to the left side.
Debug wrong parallel pages syn-	1
chronization when an \edtext	\clearl@drightpage: Use
falls accross two pages.	\newpage instead of
75	\clearpage.
v1.10.1.	81
\line@list@stuffR: Revert modi-	\ifledRcol: Remove false boolean
fication of 1.4.2, which makes	settings which are not needed.
bugs with numbering. Leave	15
	\Pages: Prevent false overfull
	hboxes when using \Pages out-
	side of pages environment.
	76

When using shiftedpstarts option, a <code>\l@ddleftbox</code> with a null height will advance the <code>\pagetotal</code> in any case.	75
astanza : Enable the use of optional argument of <code>&</code> in <code>astanza</code> environment.	63
v1.13.1.	
<code>\correct@footinsX@box</code> : Call <code>\correct@footinsX@box</code> and <code>\correct@Xfootins@box</code> directly in <code>\print@notesX@forpages</code> and <code>\print@Xnotes@forpages</code>	56
Correct <code>\correct@footinsX@box</code> and <code>\correct@Xfootins@box</code>	56
\Pages : Prevent false empty page after <code>\Pages</code> (bug added in 1.13.0)	75
v1.14.0.	
General: Fix bug with line number position when using <code>\eledsection</code> and similar commands for RTL texts with Lua ^L TeX.	1
The <code>\newifs</code> are not followed by boolean values set to false, because it is the TeX default setting.	1