

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

Maïeul Rouquette[†]based on the original **ledpar** by Peter Wilson
Herries Press[‡]

Abstract

The **eledmac** package, which is based on the PLAIN \TeX 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 v1.15.0, last revised 2015/04/13.

[†]maieul at maieul dot net

[‡]herries dot press at earthlink dot net

4 Facing pages	7
4.1 Basic usage	7
4.2 Text width	7
4.3 Setting	8
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
5.1 Environments	9
5.2 Line numbering scheme	9
5.3 chunk	10
5.4 <code>\AtEveryPstart</code> and <code>\AtEveryPstartCall</code>	10
5.5 Language setting	10
5.6 Shifting	11
6 Numbering text lines and paragraphs	11
7 Verse	12
8 Side notes	14
9 Parallel ledgroups	15
9.1 Parallel ledgroups and <code>setspace</code> package	16
10 Sectioning commands	16
11 Implementation overview	16
12 Preliminaries	17
12.1 Messages	18
13 Sectioning commands	18
14 Line counting	21
14.1 Choosing the system of lineation	21
14.2 Line-number counters and lists	24
14.3 Reading the line-list file	25
14.4 Commands within the line-list file	26
14.5 Writing to the line-list file	34
15 Marking text for notes	36
15.1 Specific hooks and commands for notes	37
15.1.1 Notes to be printed on one side only	37
15.1.2 Familiar footnotes without marks	38

<i>Contents</i>	3
15.1.3 Create hooks	39
15.1.4 Init standards series (A,B,C,D,E,Z)	39
16 Pstart numbers dumping and restoration	39
17 Parallel environments	40
18 Paragraph decomposition and reassembly	42
18.1 Boxes, counters, \pstart and \pend	42
18.2 Processing one line	47
18.3 Line and page number computation	50
18.4 Line number printing	53
18.5 Pstart number printing in side	55
18.6 Add insertions to the vertical list	57
18.7 Penalties	57
18.8 Printing leftover notes	58
19 Footnotes	59
19.1 Normal footnote formatting	59
19.2 Footnotes output specific to \Pages	59
20 Cross referencing	63
21 Side notes	64
22 Familiar footnotes	65
23 Verse	66
24 Naming macros	68
25 Counts and boxes for parallel texts	68
26 Fixing babel	70
27 Parallel columns	73
28 Parallel pages	78
29 Sections' titles' commands	90
30 Page break/no page break, depending on the specific line	91
31 Parallel ledgroup	92
32 The End	95

Appendix A Some things to do when changing version	96
Appendix A.1 Migration to <i>eledpar</i> 1.4.3	96
References	96
Index	96
Change History	107

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.

`\AtBeginPairs` 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.

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

When you use `\stanza`, the visible rule may shift when a verse has a hanging indent. To prevent shifting, use `\setstanzaindents` 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}
```

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`.

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.

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. 29). As an advisable setting:

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

4.4.2 Notes for one side only

`\onlyXside` You may want to typeset notes on one side only (either left or right). Use
`\onlysideX` `\onlyXside[⟨s⟩]{⟨p⟩}` to set critical notes, and `\onlysideX[⟨s⟩]{⟨p⟩}` to set familiar notes. `{⟨p⟩}` 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.

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

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

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

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:

```
\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

5.1 Environments

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 like-
Rightside	wise in the Rightside environment. The number of Leftside and Rightside environments must be the same.

5.2 Line numbering scheme

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.

Generally speaking, controls like `\firstlinenum` or `\linenummargin` apply to sequential and left texts. To effect right texts only, they have to be within

```
\firstlinenum
\linenumincrement
\firstsublinenum
\sublinenumincrement
\firstlinenum*
\linenumincrement*
\firstsublinenum*
\sublinenumincrement*
```

`\lineationR` a `Rightside` environment. `\lineationR` macro is the equivalent of `eledmac`
`\lineation*` `\lineation` macro for the right side. `\lineation*` macro is the equivalent of
`eledmac \lineation` macro for both sides.

5.3 chunk

`\pstart` In a serial (non-parallel) mode, each numbered paragraph, or chunk, is contained
`\pend` 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.

5.4 \AtEveryPstart and \AtEveryPstartCall

In general, remember that the moment where a `\pstart` is called is different from
the moment when the `\pstart... \pend` content is printed, which is when `\Pages`
or `\Columns` is processed.

Consequently:

- The argument of `\AtEveryPstart` (see 4.2.3 p. 14) is called before every
chunk is printed, except if you used an optional argument for the `\pstart`.
- The argument of `\AtEveryPstartCall` is called before every `\pstart`.

5.5 Language setting

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 lan-
guage 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.

5.6 Shifting

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 `\endnumbering` 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}}

```

`\numberpstarttrue` 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@@@`’ 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 be 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]{%
  \hskip -#1\llap{\textbf{#2}}\hskip #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}
  ...
\end{Leftside}
\end{pairs}
```

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

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).

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}[2015/04/13 v1.15.0 eledmac extension for parallel texts]%
4
```

Few commands use `\xspace` command.

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

With the option ‘`shiftedpstarts`’ a long `pstart` one 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` is also set TRUE if we are doing parallel pages. `\ifledRcol` is set TRUE if we 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>.nn`, where `nn` is the section number. However, for right side texts the file is called `<jobname>.nnR`. 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@sectioningRout=\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 L^AT_EX 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 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     \ledend@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     \ledend@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:

<code>\bypstart@Rtrue</code>	• line-of-page : <code>bypstart@R = false</code> and <code>bypage@R = true</code> .
<code>\bypstart@Rfalse</code>	• line-of-pstart : <code>bypstart@R = true</code> and <code>bypage@R = false</code> .

`\ifbypage@R`

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

`\bypage@Rfalse`

```

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       \unless\ifnocritical@%
147         \pstartinfootnote[] [false]%
148       \fi%
149     \else
150       \def\@tempb{pstart}%
151       \ifx\@tempa\@tempb
152         \global\bypage@Rfalse
153         \global\bypstart@Rtrue
154         \unless\ifnocritical@%
155           \pstartinfootnote%
156         \fi%
157       \else
158         \def\@tempb{section}%
159         \ifx\@tempa\@tempb
160           \global\bypage@Rfalse%
161           \global\bypstart@Rfalse%
162           \unless\ifnocritical@%
163             \pstartinfootnote[] [false]%
164           \fi%
165         \else
166           \led@warn@BadLineation
167         \fi%
168       \fi
169     \fi
170 \fi}}

```

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

```

171 \WithSuffix\newcommand\lineation*[1]{%
172   \lineation{#1}%
173   \lineationR{#1}%
174 }%

```

`\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.

```

175 \newcount\line@marginR
176 \renewcommand*{\linenummargin}[1]{%
177   \l@getline@margin{#1}%
178   \ifnum\@l@dtmpcntb>\m@ne
179     \ifledRcol
180       \global\line@marginR=\@l@dtmpcntb
181     \else
182       \global\line@margin=\@l@dtmpcntb
183     \fi
184   \fi}}

```

By default put right text numbers at the right.

```

185 \line@marginR=\@ne
186

```

`\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.

```

187 \newcounter{firstlinenumR}
188 \setcounter{firstlinenumR}{5}
189 \newcounter{linenumincrementR}
190 \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.

```

191 \newcounter{firstsublinenumR}
192 \setcounter{firstsublinenumR}{5}
193 \newcounter{sublinenumincrementR}
194 \setcounter{sublinenumincrementR}{5}
195

```

`\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 196 \providecommand*{\firstlinenum}{}
\firstlinenum*       197 \providecommand*{\linenumincrement}{}
\linenumincrement*   198 \providecommand*{\firstsublinenum}{}
\firstsublinenum*    199 \providecommand*{\sublinenumincrement}{}
\sublinenumincrement* 200 \renewcommand*{\firstlinenum}[1]{%
201   \ifledRcol \setcounter{firstlinenumR}{#1}%
202   \else      \setcounter{firstlinenumR}{#1}%
203   \fi}
204 \renewcommand*{\linenumincrement}[1]{%
205   \ifledRcol \setcounter{linenumincrementR}{#1}%

```

```

206 \else \setcounter{linenumincrement}{#1}%
207 \fi}
208 \renewcommand*{\firstsublinenum}[1]{%
209 \ifledRcol \setcounter{firstsublinenumR}{#1}%
210 \else \setcounter{firstsublinenum}{#1}%
211 \fi}
212 \renewcommand*{\sublinenumincrement}[1]{%
213 \ifledRcol \setcounter{sublinenumincrementR}{#1}%
214 \else \setcounter{sublinenumincrement}{#1}%
215 \fi}
216 \WithSuffix\newcommand\firstlinenum*[1]{\setcounter{firstlinenumR}{#1}\setcounter{firstlinenum}{#1}}
217 \WithSuffix\newcommand\linenumincrement*[1]{\setcounter{linenumincrementR}{#1}\setcounter{linenumincrement}{#1}}
218 \WithSuffix\newcommand\firstsublinenum*[1]{\setcounter{subfirstlinenumR}{#1}\setcounter{subfirstlinenum}{#1}}
219 \WithSuffix\newcommand\sublinenumincrement*[1]{\setcounter{sublinenumincrementR}{#1}\setcounter{sublinenumincrement}{#1}}

```

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

```

220 \newcommand*{\Rlineflag}{R}
221

```

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

```

222 \newcommand*{\linenumrepR}[1]{\@arabic{#1}}
223 \newcommand*{\sublinenumrepR}[1]{\@arabic{#1}}
224

```

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

```

225 \newcommand*{\leftlinenumR}{%
226 \l@dlinenumR
227 \kern\linenumsep}
228 \newcommand*{\rightlinenumR}{%
229 \kern\linenumsep
230 \l@dlinenumR}
231 \newcommand*{\l@dlinenumR}{%
232 \numlabfont\linenumrepR{\line@numR}\Rlineflag%
233 \ifsublines@
234 \ifnum\subline@num>\z@
235 \unskip\fullstop\sublinenumrepR{\subline@numR}%
236 \fi
237 \fi}
238

```

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.

```

239 \newcount\line@numR
240 \newcount\subline@numR
241 \newcount\absline@numR
242

```

`\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.

`\insertlines@listR`

`\actionlines@listR`

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

```

243 \list@create{\line@listR}
244 \list@create{\insertlines@listR}
245 \list@create{\actionlines@listR}
246 \list@create{\actions@listR}
247 \list@create{\sw@listR}%
248 \list@create{\sw@list@inedtextR}%
249

```

`\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`

```

250 \list@create{\linesinpar@listL}
251 \list@create{\linesinpar@listR}
252 \list@create{\maxlinesinpar@list}
253

```

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

```

254 \newcount\page@numR
255

```

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.

```

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

```

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

```

257   \ifledRcol
258     \list@clear{\line@listR}%
259     \list@clear{\insertlines@listR}%
260     \list@clear{\actionlines@listR}%
261     \list@clear{\actions@listR}%
262     \list@clear{\linesinpar@listR}%

```

```

263 \list@clear{\linesonpage@listR}
264 \list@clear{\sw@listR}%
265 \list@clear{\sw@list@inedtextR}%
266 \else
267 \list@clearing@reg
268 \list@clear{\linesinpar@listL}%
269 \list@clear{\linesonpage@listL}%
270 \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).

```

271 \list@clear{\maxlinesinpar@list}

```

Now get the file and interpret it.

```

272 \get@linelistfile{#1}%
273 \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.

```

274 \ifledRcol
275 \global\page@numR=\m@ne
276 \ifx\actionlines@listR\empty
277 \gdef\next@actionlineR{1000000}%
278 \else
279 \gl@p\actionlines@listR\to\next@actionlineR
280 \gl@p\actions@listR\to\next@actionR
281 \fi
282 \else
283 \global\page@num=\m@ne
284 \ifx\actionlines@list\empty
285 \gdef\next@actionline{1000000}%
286 \else
287 \gl@p\actionlines@list\to\next@actionline
288 \gl@p\actions@list\to\next@action
289 \fi
290 \fi}
291

```

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 what it does depends on whether right text is being processed.

```

292 \newcommand{\@nl@regR}{%
293   \ifx\l@dchset@num\relax \else
294     \advance\absline@numR \@ne
295     \set@line@action
296     \let\l@dchset@num\relax
297     \advance\absline@numR \m@ne
298     \advance\line@numR \m@ne%    % do we need this?
299   \fi
300   \advance\absline@numR \@ne
301   \ifx\next@page@numR\relax \else
302     \page@action
303     \let\next@page@numR\relax
304   \fi
305   \ifx\sub@change\relax \else
306     \ifnum\sub@change>\z@
307       \sublines@true
308     \else
309       \sublines@false
310     \fi
311     \sub@action
312     \let\sub@change\relax
313   \fi
314   \ifcase\@lockR
315   \or
316     \@lockR \tw@
317   \or\or
318     \@lockR \z@
319   \fi
320   \ifcase\sub@lockR
321   \or
322     \sub@lockR \tw@
323   \or\or
324     \sub@lockR \z@
325   \fi
326   \ifsublines@
327     \ifnum\sub@lockR<\tw@
328       \advance\subline@numR \@ne
329     \fi
330   \else
331     \ifnum\@lockR<\tw@
332       \advance\line@numR \@ne \subline@numR \z@
333     \fi
334   \fi}
335
336 \renewcommand*{\@nl}[2]{%
```

```

337 \fix@page{#1}%
338 \ifledRcol
339 \@nl@regR
340 \else
341 \@nl@reg
342 \fi}
343

```

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

```

\fix@page 344 \newcount\last@page@numR
345 \last@page@numR=-10000
346 \renewcommand*{\fix@page}[1]{%
347 \ifledRcol
348 \ifnum #1=\last@page@numR
349 \else
350 \ifbypage@R
351 \line@numR \z@ \subline@numR \z@
352 \fi
353 \page@numR=#1\relax
354 \last@page@numR=#1\relax
355 \def\next@page@numR{#1}%
356 \fi
357 \else
358 \ifnum #1=\last@page@num
359 \else
360 \ifbypage@
361 \line@num \z@ \subline@num \z@
362 \fi
363 \page@num=#1\relax
364 \last@page@num=#1\relax
365 \def\next@page@num{#1}%
366 \listxadd{\normal@page@break}{\the\absline@num}
367 \fi
368 \fi}
369

```

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

```

370 \renewcommand*{\@adv}[1]{%
371 \ifsublines@
372 \ifledRcol
373 \advance\subline@numR by #1\relax
374 \ifnum\subline@numR<\z@
375 \led@warn@BadAdvancelineSubline
376 \subline@numR \z@
377 \fi
378 \else
379 \advance\subline@num by #1\relax
380 \ifnum\subline@num<\z@

```

```

381      \led@warn@BadAdvancelineSubline
382      \subline@num \z@
383      \fi
384  \fi
385  \else
386    \ifledRcol
387      \advance\line@numR by #1\relax
388      \ifnum\line@numR<\z@
389        \led@warn@BadAdvancelineLine
390        \line@numR \z@
391      \fi
392    \else
393      \advance\line@num by #1\relax
394      \ifnum\line@num<\z@
395        \led@warn@BadAdvancelineLine
396        \line@num \z@
397      \fi
398    \fi
399  \fi
400  \set@line@action}
401

```

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

```

402 \renewcommand*{\@set}[1]{%
403   \ifledRcol
404     \ifsublines@
405       \subline@numR=#1\relax
406     \else
407       \line@numR=#1\relax
408     \fi
409     \set@line@action
410   \else
411     \ifsublines@
412       \subline@num=#1\relax
413     \else
414       \line@num=#1\relax
415     \fi
416     \set@line@action
417   \fi}
418

```

\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.

```

419 \renewcommand*{\l@d@set}[1]{%
420   \ifledRcol
421     \line@numR=#1\relax

```

```

422     \advance\line@numR \@ne
423     \def\l@dchset@num{#1}
424   \else
425     \line@num=#1\relax
426     \advance\line@num \@ne
427     \def\l@dchset@num{#1}
428   \fi}
429 \let\l@dchset@num\relax
430

```

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

```

431 \renewcommand*{\page@action}{%
432   \ifledRcol
433     \xright@appenditem{\the\absline@numR}\to\actionlines@listR
434     \xright@appenditem{\next@page@numR}\to\actions@listR
435   \else
436     \xright@appenditem{\the\absline@num}\to\actionlines@list
437     \xright@appenditem{\next@page@num}\to\actions@list
438   \fi}

```

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

```

439 \renewcommand*{\set@line@action}{%
440   \ifledRcol
441     \xright@appenditem{\the\absline@numR}\to\actionlines@listR
442     \ifsublines@
443       \l@dtempcnta=-\subline@numR
444     \else
445       \l@dtempcnta=-\line@numR
446     \fi
447     \advance\l@dtempcnta by -5000\relax
448     \xright@appenditem{\the\l@dtempcnta}\to\actions@listR
449   \else
450     \xright@appenditem{\the\absline@num}\to\actionlines@list
451     \ifsublines@
452       \l@dtempcnta=-\subline@num
453     \else
454       \l@dtempcnta=-\line@num
455     \fi
456     \advance\l@dtempcnta by -5000\relax
457     \xright@appenditem{\the\l@dtempcnta}\to\actions@list
458   \fi}
459

```

`\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.

```

460 \renewcommand*{\sub@action}{%
461   \ifledRcol
462     \xright@appenditem{\the\absline@numR}\to\actionlines@listR

```

```

463 \ifsublines@
464 \xright@appenditem{-1001}\to\actions@listR
465 \else
466 \xright@appenditem{-1002}\to\actions@listR
467 \fi
468 \else
469 \xright@appenditem{\the\absline@num}\to\actionlines@list
470 \ifsublines@
471 \xright@appenditem{-1001}\to\actions@list
472 \else
473 \xright@appenditem{-1002}\to\actions@list
474 \fi
475 \fi}
476

```

`\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.

```

477 \newcount\@lockR
478 \newcount\sub@lockR
479
480 \newcommand*{\do@lockonR}{%
481 \xright@appenditem{\the\absline@numR}\to\actionlines@listR
482 \ifsublines@
483 \xright@appenditem{-1005}\to\actions@listR
484 \ifnum\sub@lockR=\z@
485 \sub@lockR \@ne
486 \else
487 \ifnum\sub@lockR=\thr@@
488 \sub@lockR \@ne
489 \fi
490 \fi
491 \else
492 \xright@appenditem{-1003}\to\actions@listR
493 \ifnum\@lockR=\z@
494 \@lockR \@ne
495 \else
496 \ifnum\@lockR=\thr@@
497 \@lockR \@ne
498 \fi
499 \fi
500 \fi}
501
502 \renewcommand*{\do@lockon}{%
503 \ifx\next\lock@off
504 \global\let\lock@off=\skip@lockoff
505 \else
506 \ifledRcol
507 \do@lockonR
508 \else

```

```

509      \do@lockonL
510      \fi
511    \fi}

\lock@off \lock@off adds an entry to the action-code list to turn line number locking off.
\do@lockoff 512
\do@lockoffR 513
\skip@lockoff 514 \newcommand{\do@lockoffR}{%
515   \xright@appenditem{\the\absline@numR}\to\actionlines@listR
516   \ifsublines@
517     \xright@appenditem{-1006}\to\actions@listR
518     \ifnum\sub@lockR=\tw@
519       \sub@lockR \thr@@
520     \else
521       \sub@lockR \z@
522     \fi
523   \else
524     \xright@appenditem{-1004}\to\actions@listR
525     \ifnum\@lockR=\tw@
526       \@lockR \thr@@
527     \else
528       \@lockR \z@
529     \fi
530   \fi}
531
532 \renewcommand*{\do@lockoff}{%
533   \ifledRcol
534     \do@lockoffR
535   \else
536     \do@lockoffL
537   \fi}
538 \global\let\lock@off=\do@lockoff
539
\n@num

```

\@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 analogously 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@R\global\line@numR=0\fi
577   \xright@appenditem{#1}\to\linesinpar@listR}
578

```

`\@lopL` `\@lopL{<num>}` adds its argument to the `\linesonpage@listL` list, and analogously `\@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 another file that we keep open.

```

\first@linenum@out@Rtrue
\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
`\flag@end` send the `\@ref` command to the line-list file.

`\startsub` `\startsub` and `\endsub` turn sub-lineation on and off, by writing appropriate
`\endsub` 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.

```
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`

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`.

```

649 \renewcommand*{\set@line}{%
650   \ifledRcol
651     \ifx\line@listR\empty
652       \global\noteschanged@true
653       \xdef\l@d@nums{000|000|000|000|000|000|000|000|\edfont@info}%
654     \else
655       \gl@p\line@listR\to\@tempb
656       \xdef\l@d@nums{\@tempb|\edfont@info}%
657       \global\let\@tempb=\undefined
658     \fi
659   \else
660     \ifx\line@list\empty
661       \global\noteschanged@true
662       \xdef\l@d@nums{000|000|000|000|000|000|000|000|\edfont@info}%
663     \else
664       \gl@p\line@list\to\@tempb
665       \xdef\l@d@nums{\@tempb|\edfont@info}%
666       \global\let\@tempb=\undefined
667     \fi
668   \fi}
669
```

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`

```

670 \newcommand{\newseries@eledpar}[1]{%

```

15.1.1 Notes to be printed on one side only

eledpar allows notes to be printed on one side only. We need to declare these options. We also need boolean flags, and to set them to true when a note series is not printed on one side. We check the `nofamiliar` and `nocritical` eledmac options.

```

671   \unless\ifnocritical{%
672     \csgdef{onlysideX@#1}{}%
673     \global\newbool{keepforXside@#1}%
674   \fi%
675   \unless\ifnofamiliar{%
676     \global\newbool{keepforsideX@#1}%
677     \csgdef{onlyXside@#1}{}%
678   \fi%

```

15.1.2 Familiar footnotes without marks

The `\footnoteXnomk` commands are for notes which are printed on 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.

First, check the `nofamiliar` option of `eledmac`

```
679 \unless\ifnofamiliar%
680 % So declare the list.
681 % \begin{macrocode}
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 tools for familiar notes without marks

```
703 \fi
End of \newseries@eledpar.
704 }%
```

15.1.3 Create hooks

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

```
705 \unless\ifnocritical%
706 \newhookcommand@series{onlyXside}%
707 \fi%
708 \unless\ifnofamiliar%
709 \newhookcommand@series{onlysideX}%
710 \fi
711
712
```

15.1.4 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.

```
713 \newcommand{\init@series@eledpar}{%
714 \def\do##1{\newseries@eledpar{##1}}%
715 \dolistloop{\@series}%
716 }%
717 \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
718 \list@create{\list@pstartL@pc}%
719 \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 720 \def\dump@pstartL@pc{%
721   \xright@appenditem{\the\c@pstartL}\to\list@pstartL@pc%
722   \global\cslet{numberpstart@L\the\l@dnumpstartsL}{\ifnumberpstart}%
723 }%
724
725 \def\dump@pstartR@pc{%
726   \xright@appenditem{\the\c@pstartR}\to\list@pstartR@pc%
727   \global\cslet{numberpstart@R\the\l@dnumpstartsR}{\ifnumberpstart}%
728 }%
729

\restore@pstartL@pc And so, the commands to restore them
\restore@pstartR@pc 730 \def\restore@pstartL@pc{%
731   \ifx\list@pstartL@pc\empty\else%
732     \gl@p\list@pstartL@pc\to\@temp%
733     \global\c@pstartL=\@temp%
734   \fi%
735 }%
736 \def\restore@pstartR@pc{%
737   \ifx\list@pstartR@pc\empty\else%
738     \gl@p\list@pstartR@pc\to\@temp%
739     \global\c@pstartR=\@temp%
740   \fi%
741 }%

```

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
pages parallel pages.
chapterinpages 742 \newenvironment{pairs}{%}
743   \l@dpairingtrue
744   \l@dpagingfalse
745   \initnumbering@sectcmd
746   \at@begin@pairs%
747 }{%
748   \l@dpairingfalse
749 }
750

\AtBeginPairs The \AtBeginPairs macro just define a \at@begin@pairs macro, called at the
beginning of each pairs environments.
751 \newcommand{\AtBeginPairs}[1]{\xdef\at@begin@pairs{#1}}%
752 \def\at@begin@pairs{%
753

```


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.

```

754 \newenvironment{pages}{%
755   \let\oldchapter\chapter
756   \let\chapter\chapterinpages
757   \l@dpairingtrue
758   \l@dpagingtrue
759   \initnumbering@sectcmd
760   \setlength{\Lcolwidth}{\textwidth}%
761   \setlength{\Rcolwidth}{\textwidth}%
762 }{%
763   \l@dpairingfalse
764   \l@dpagingfalse
765   \let\chapter\oldchapter
766 }
767 \newcommand{\chapterinpages}{\thispagestyle{plain}%
768                               \global\@topnum\z@
769                               \@afterindentfalse
770                               \secdef\@chapter\@schapter}
771

```

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

```

772 \newif\ifinstanzaL
773 \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.

```

774 \newenvironment{Leftside}{%
775   \ledRcolfalse
776   \setcounter{pstartL}{1}
777   \let\pstart\pstartL
778   \let\thepstart\thepstartL
779   \let\pend\pendL
780   \let\memorydump\memorydumpL
781   \Leftsidehook
782   \let\old@startstanza\@startstanza
783   \def\@startstanza[##1]{\global\instanzaLtrue\old@startstanza[##1]}
784 }{
785   \Leftsidehookend}

```

`\Leftsidehook` Hooks into the start and end of the `Leftside` and `Rightside` environments. These
`\Leftsidehookend` are initially empty.

```

\Rightsidehook 786 \newcommand*{\Leftsidehook}{}
\Rightsidehookend

```

```

787 \newcommand*{\Leftsidehookend}{}
788 \newcommand*{\Rightsidehook}{}
789 \newcommand*{\Rightsidehookend}{}
790

```

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.

```

791 \newenvironment{Rightside}{%
792   \ledRcoltrue
793   \let\beginnumbering\beginnumberingR
794   \let\endnumbering\endnumberingR
795   \let\pausenumbering\pausenumberingR
796   \let\resumenumbering\resumenumberingR
797   \let\memorydump\memorydumpR
798   \let\thepstart\thepstartR
799   \let\pstart\pstartR
800   \let\pend\pendR
801   \let\ledpb\ledpbR
802   \let\lednopb\lednopbR
803   \let\lineation\lineationR
804   \Rightsidehook
805   \let\old@startstanza\@startstanza
806   \def\@startstanza[##1]{\global\instanzaRtrue\old@startstanza[##1]}
807 }{%
808   \ledRcolfalse
809   \Rightsidehookend
810 }
811

```

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.

```
812 \newcount\num@linesR
813 \newbox\one@lineR
814 \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.

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.

```
815
816 \newcounter{pstartL}
817 \renewcommand{\thepstartL}{\bfseries\@arabic\c@pstartL}. }
818 \newcounter{pstartR}
819 \renewcommand{\thepstartR}{\bfseries\@arabic\c@pstartR}. }
820
821 \newcommand*{\pstartL}[1][1]{%
822   \if@nobreak%
823     \let\@oldnobreak\@nobreaktrue%
824   \else%
825     \let\@oldnobreak\@nobreakfalse%
826   \fi%
827   \@nobreaktrue%
828   \ifluatex%
829     \xdef\l@luatextextdir@L{\the\luatextextdir}%
830     \xdef\l@luatexpardir@L{\the\luatexpardir}%
831     \xdef\l@luatexbodydir@L{\the\luatexbodydir}%
832   \fi%
833   \ifnumbering \else%
834     \led@err@PstartNotNumbered%
835     \beginnumbering%
836   \fi%
837   \ifnumberedpar@%
838     \led@err@PstartInPstart%
839   \pend%
840   \fi%
```

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

```
841 \ifpst@rtedL\else%
842   \list@clear{\inserts@list}%
843   \global\let\next@insert=\empty%
```

```

844 \global\pst@rtedLtrue%
845 \fi%
846 \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.
847 \global\advance\l@dnumpstartsL \@ne%
848 \ifnum\l@dnumpstartsL>\l@dc@maxchunks%
849 \led@err@TooManyPstarts%
850 \global\l@dnumpstartsL=\l@dc@maxchunks%
851 \fi%
852 \global\setnamebox{\l@dLcolrawbox\the\l@dnumpstartsL}=\vbox\bgroup%
853 \ifautopar\else%
854 \ifnumberpstart%
855 \ifsidepstartnum%
856 \else%
857 \thepstartL%
858 \fi%
859 \fi%
860 \fi%
861 \hsize=\lcolwidth%
862 \numberedpar@true%
863 \iflabelpstart\protected@edef\@currentlabel%
864 {\p@pstartL\thepstartL}\fi%
    Dump the optional arguments
865 \ifstrempy{#1}%
866 {\csgdef{before@pstartL@the\l@dnumpstartsL}{\at@every@pstart}}%
867 {\csgdef{before@pstartL@the\l@dnumpstartsL}{\noindent#1}}%
868 \at@every@pstart@call%
869 }
870 \newcommandx*{\pstartR}[1][1]{%
871 \if@nbreak%
872 \let\@oldnbreak\@nbreaktrue%
873 \else%
874 \let\@oldnbreak\@nbreakfalse%
875 \fi%
876 \@nbreaktrue%
877 \ifluatex%
878 \xdef\l@luatextextdir@R{\the\luatextextdir}%
879 \xdef\l@luatexpardir@R{\the\luatexpardir}%
880 \xdef\l@luatexbodydir@R{\the\luatexbodydir}%
881 \fi%
882 \ifnumberingR \else%
883 \led@err@PstartNotNumbered%
884 \beginnumberingR%
885 \fi%
886 \ifnumberedpar@%
887 \led@err@PstartInPstart%
888 \pendR%

```

```

889 \fi%
890 \ifpstart@rtedR\else%
891   \list@clear{\inserts@listR}%
892   \global\let\next@insertR=\empty%
893   \global\pstart@rtedRtrue%
894 \fi%
895 \begingroup\normal@pars%
896 \global\advance\l@dnumpstartsR \@one%
897 \ifnum\l@dnumpstartsR>\l@dc@maxchunks%
898   \led@err@TooManyPstarts%
899   \global\l@dnumpstartsR=\l@dc@maxchunks%
900 \fi%
901 \global\setnamebox{\l@drColrawbox\the\l@dnumpstartsR}=\vbox\bgroup%
902   \ifautopar\else%
903     \ifnumberpstart%
904       \ifsidepstartnum\else%
905         \thepstartR%
906       \fi%
907     \fi%
908   \fi%
909 \hsize=\Rcolwidth%
910 \numberedpar@true%
911 \iflabelpstart\protected@edef\@currentlabel%
912   {\p@pstartR\thepstartR}\fi%
913 \ifstrempy{#1}%
914   {\csgdef{before@pstartR@the\l@dnumpstartsR}{\at@every@pstart}}%
915   {\csgdef{before@pstartR@the\l@dnumpstartsR}{\noindent#1}}%
916 \at@every@pstart@call%
917 }

```

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

```

918 \newcommandx*{\pendL}[1][1]{%
919   \ifnumbering \else%
920     \led@err@PendNotNumbered%
921   \fi%
922   \ifnumberedpar@ \else%
923     \led@err@PendNoPstart%
924   \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.

```

925 \l@dzeropenalties%
926 \endgraf\global\num@lines=\prevgraf\egroup%
927 \global\par@line=0%

```

End the group that was begun in the `\pstart`.

```

928 \endgroup%

```

```

929 \ignorespaces%
930 \@oldnobreak%
931 \dump@pstartL@pc%
932 \ifnumberpstart%
933   \addtocounter{pstartL}{1}%
934 \fi
935 \parledgroup@beforenotes@save{L}%
  Dump content of the optional argument.
936 \ifstrempy{#1}%
937   {\csgdef{after@pendL@the\l@dnumpstartsL}{\at@every@pend}}%
938   {\csgdef{after@pendL@the\l@dnumpstartsL}{\noindent#1}}%
939 }

```

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

```

940 \newcommand*{\pendR}[1][1]{%
941   \ifnumberingR \else%
942     \led@err@PendNotNumbered%
943   \fi%
944   \ifnumberedpar@ \else%
945     \led@err@PendNoPstart%
946   \fi%
947   \l@dzeropenalties%
948   \endgraf\global\num@linesR=\prevgraf\egroup%
949   \global\par@lineR=0%
950   \endgroup%
951   \ignorespaces%
952   \@oldnobreak%
953   \dump@pstartR@pc%
954   \ifnumberpstart%
955     \addtocounter{pstartR}{1}%
956   \fi%
957   \parledgroup@beforenotes@save{R}%
958   \ifstrempy{#1}%
959     {\csgdef{after@pendR@the\l@dnumpstartsR}{\at@every@pend}}%
960     {\csgdef{after@pendR@the\l@dnumpstartsR}{\noindent#1}}%
961 }
962

```

`\AtEveryPstartCall` The `\AtEveryPstartCall` argument is called when the `\pstartL` or `\pstartR` is called. That is different of `\AtEveryPstart` the argument of which is called when the `\pstarts` are printed.

```

963 \newcommand{\AtEveryPstartCall}[1]{\xdef\at@every@pstart@call{\unexpanded{#1}}}%
964 \gdef\at@every@pstart@call{}%

```

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

```

965 \newif\ifprint@last@after@pendL%
966 \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 analagously for a line
`\l@drightbox` of right text.

```
967 \newbox\l@dleftbox
968 \newbox\l@drightbox
969
```

`\countLline` We need to know the number of lines processed.

```
\countRline 970 \newcount\countLline
971 \countLline \z@
972 \newcount\countRline
973 \countRline \z@
974
```

`\@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 975 \newcount\@donereallinesL
976 \newcount\@donetotallinesL
977 \newcount\@donereallinesR
978 \newcount\@donetotallinesR
979
```

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

```
980 \newcommand*\do@lineL{%
981 \letcs{\ifnumberpstart}{numberpstart@L\the\l@dpscl}%
982 \advance\countLline \@ne%
983 \ifvbox\namebox{\l@dLcolrawbox\the\l@dpscl}%
984 {\vbadness=10000%
985 \splittopskip=\z@%
986 \do@lineLhook%
987 \l@emptyd@ta%
988 \global\setbox\one@line=\vsplit\namebox{\l@dLcolrawbox\the\l@dpscl}%
989 to\baselineskip}%
990 \IfStrEq{\splitfirstmarks\parledgroup@}{begin}{\parledgroup@notes@startL}{}%
991 \unvbox\one@line \global\setbox\one@line=\lastbox%
992 \getline@numL%
993 \ifnum\@lock>\@ne%
994 \inserthangingsymboltrue%
995 \else%
996 \inserthangingsymbolfalse%
997 \fi%
998 \setbox\l@dleftbox%
```

```

999 \hb@xt@ \Lcolwidth{%
1000 \ifl@dhidenumber%
1001 \global\l@dhidenumberfalse%
1002 \f@x@l@cks%
1003 \else%
1004 \affixline@num%
1005 \fi%
1006 \xifinlist{\the\l@dpscL}{\eled@sections@}%
1007 {\add@inserts\affixside@note}%
1008 {\print@lineL}}%
1009 \add@penaltiesL%
1010 \global\advance\@donereallinesL\@ne%
1011 \global\advance\@donetotallinesL\@ne%
1012 \else%
1013 \setbox\l@dleftbox \hb@xt@ \Lcolwidth{\hspace*\Lcolwidth}}%
1014 \global\advance\@donetotallinesL\@ne%
1015 \fi}
1016
1017

```

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

```

1018 \def\print@lineL{%
1019 \affixpstart@numL%
1020 \l@dld@ta %space kept for backward compatibility
1021 \add@inserts\affixside@note%
1022 \l@dlsn@te %space kept for backward compatibility
1023 {\ledllfill\hb@xt@ \Lcolwidth{%
1024 \do@insidelineLhook%
1025 \ifluatex%
1026 \luatextextdir\l@luatextextdir@L%
1027 \fi%
1028 \new@lineL%
1029 \inserthangingsymbolL%
1030 \l@dunhbox@line{\one@line}}\ledrlfill\l@drd@ta%
1031 \l@drrsn@te}}
1032

```

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

```

1033 \def\print@eledsectionL{%
1034 \addtocounter{pstartL}{-1}%
1035 \ifdefstring{\@eledsectnotoc}{L}{\ledsectnotoc}{%
1036 \ifdefstring{\@eledsectmark}{L}{\ledsectnomark}%
1037 \numdef{\temp@}{\l@dpscL-1}%
1038 \xifinlist{\temp@}{\eled@sections@}{\@nobreaktrue}{\@nobreakfalse}%
1039 \@eled@sectioningtrue%
1040 \bgroup%
1041 \ifluatex%
1042 \luatextextdir\l@luatextextdir@L%
1043 \luatexpardir\l@luatexpardir@L%

```



```

1044      \luatexbodydir\l@luatexbodydir@L%
1045      \ifdefstring{\l@luatextextdir@L}{TRT}{\@RTLtrue}{}%
1046      \fi%
1047      \csuse{eled@sectioning@the\l@dpscl}%
1048      \egroup%
1049      \@eled@sectioningfalse%
1050      \global\csundef{eled@sectioning@the\l@dpscl}%
1051      \ifRTL%
1052        \hspace{-3\paperwidth}%
1053        {\hbox{\l@dunhbox@line{\one@line}} \new@line}%
1054      \else%
1055        \hspace{3\paperwidth}%
1056        {\new@line \hbox{\l@dunhbox@line{\one@line}}}%
1057      \fi%
1058      \vskip\eledsection@correcting@skip%
1059 }
1060

```

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

```

\doinlinedlineLhook 1061 \newcommand*\dolineLhook[1]{\gdef\do@lineLhook{#1}}%
\doinlinedlineRhook 1062 \newcommand*\dolineRhook[1]{\gdef\do@lineRhook{#1}}%
1063 \newcommand*\doinlinedlineLhook[1]{\gdef\do@insidelineLhook{#1}}%
1064 \newcommand*\doinlinedlineRhook[1]{\gdef\do@insidelineRhook{#1}}%
1065

```

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

```

\dolineRhook 1066 \newcommand*\dolineRhook{}
\doinlinedlineLhook 1067 \newcommand*\dolineRhook{}
\doinlinedlineRhook 1068 \newcommand*\dolineinsidelineLhook{}
1069 \newcommand*\dolineinsidelineRhook{}
1070

```

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

```

1071 \newcommand*\do@lineR{%
1072   \letcs{\ifnumberpstart}{numberpstart@R\the\l@dpscl}%
1073   \ledRcol@true%
1074   \advance\countRline \@ne%
1075   \ifvbox\namebox{\l@dRcolrawbox\the\l@dpscl}%
1076   {\vbadness=10000%
1077    \splittopskip=\z@%
1078    \do@lineRhook%
1079    \l@emptyd@ta%
1080    \global\setbox\one@lineR=\vsplit\namebox{\l@dRcolrawbox\the\l@dpscl}%
1081    to\baselineskip}%
1082   \IfStrEq{\splitfirstmarks\parledgroup@}{begin}{\parledgroup@notes@startR}{}%
1083   \unvbox\one@lineR \global\setbox\one@lineR=\lastbox%

```

```

1084 \getline@numR%
1085 \ifnum\@lockR>\@ne%
1086   \inserthangingsymbolRtrue%
1087 \else%
1088   \inserthangingsymbolRfalse%
1089 \fi%
1090 \setbox\l@drightbox%
1091 \hb@xt@ \Rcolwidth{%
1092   \ifl@dhidenumber%
1093     \global\l@dhidenumberfalse%
1094     \f@x@l@cksR%
1095   \else%
1096     \affixline@numR%
1097   \fi%
1098   \xifinlist{\the\l@dpscR}{\eled@sectionsR@@}%
1099   {\add@insertsR\affixside@noteR}%
1100   {\print@lineR}%
1101 }%
1102 \add@penaltiesR%
1103 \global\advance\@donereallinesR\@ne%
1104 \global\advance\@donetotallinesR\@ne%
1105 \else%
1106   \setbox\l@drightbox \hb@xt@ \Rcolwidth{\hspace*{\Rcolwidth}}%
1107   \global\advance\@donetotallinesR\@ne%
1108 \fi%
1109 \ledRcol@false%
1110 }
1111
1112

```

```

\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.

```

1113 \newcommand*{\getline@numR}{%
1114   \global\advance\absline@numR \@ne
1115   \do@actionsR
1116   \do@ballastR
1117   \ifledgroupnotesR\else
1118     \ifnumberline
1119       \ifsublines@
1120         \ifnum\sub@lockR<\tw@
1121           \global\advance\subline@numR \@ne
1122         \fi
1123       \else
1124         \ifnum\@lockR<\tw@
1125           \global\advance\line@numR \@ne

```

```

1126         \global\subline@numR \z@
1127         \fi
1128     \fi
1129 \fi
1130 \fi
1131 }
1132 \newcommand*{\getline@numL}{%
1133     \global\advance\absline@num \@ne
1134     \do@actions
1135     \do@ballast
1136     \iflfdgroupnotesL@ \else
1137         \ifnumberline
1138             \ifsublines@
1139                 \ifnum\sub@lock<\tw@
1140                     \global\advance\subline@num \@ne
1141                 \fi
1142             \else
1143                 \ifnum\@lock<\tw@
1144                     \global\advance\line@num \@ne
1145                     \global\subline@num \z@
1146                 \fi
1147             \fi
1148         \fi
1149     \fi
1150 }
1151
1152

```

`\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.

```

1153 \newcommand*{\do@ballastR}{\global\ballast@count=\z@
1154     \begingroup
1155         \advance\absline@numR \@ne
1156         \ifnum\next@actionlineR=\absline@numR
1157             \ifnum\next@actionR>-1001
1158                 \global\advance\ballast@count by -\c@ballast
1159             \fi
1160         \fi
1161     \endgroup}

```

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

```

1162
1163 \newif\ifl@dskipversenumberR
1164 \newcommand*{\do@actions@fixedcodeR}{%
1165     \ifcase\l@dttempcnta%

```

```

1166 \or% % 1001
1167 \global\sublines@true
1168 \or% % 1002
1169 \global\sublines@false
1170 \or% % 1003
1171 \global\@lockR=\@ne
1172 \or% % 1004%
1173 \ifnum\@lockR=\tw@
1174 \global\@lockR=\thr@@
1175 \else
1176 \global\@lockR=\z@
1177 \fi
1178 \or% % 1005
1179 \global\sub@lockR=\@ne
1180 \or% % 1006
1181 \ifnum\sub@lockR=\tw@
1182 \global\sub@lockR=\thr@@
1183 \else
1184 \global\sub@lockR=\z@
1185 \fi
1186 \or% % 1007
1187 \l@dskipnumbertrue
1188 \or% % 1008
1189 \l@dskipversenumberRtrue%
1190 \or% % 1009
1191 \l@dhiddenumbertrue%
1192 \else%
1193 \led@warn@BadAction
1194 \fi%
1195 }
1196
1197
1198 \newcommand*{\do@actionsR}{%
1199 \global\let\do@actions@nextR=\relax
1200 \@l@dttempcntb=\absline@numR
1201 \ifnum\@l@dttempcntb<\next@actionlineR\else
1202 \ifnum\next@actionR>-1001\relax
1203 \global\page@numR=\next@actionR
1204 \ifbypage@R
1205 \global\line@numR \z@ \global\subline@numR \z@
1206 \fi
1207 \else
1208 \ifnum\next@actionR<-4999\relax % 9/05 added relax here
1209 \@l@dttempcnta=-\next@actionR
1210 \advance\@l@dttempcnta by -5001\relax
1211 \ifsublines@
1212 \global\subline@numR=\@l@dttempcnta
1213 \else
1214 \global\line@numR=\@l@dttempcnta
1215 \fi

```

```

1216     \else
1217         \@l@dttempcnta=-\next@actionR
1218         \advance\@l@dttempcnta by -1000\relax
1219         \do@actions@fixedcodeR
1220     \fi
1221 \fi
1222 \ifx\actionlines@listR\empty
1223     \gdef\next@actionlineR{1000000}%
1224 \else
1225     \gl@p\actionlines@listR\to\next@actionlineR
1226     \gl@p\actions@listR\to\next@actionR
1227     \global\let\do@actions@nextR=\do@actionsR
1228 \fi
1229 \fi
1230 \do@actions@nextR}
1231

```

18.4 Line number printing

\l@dcalcnnum \affixline@numR is the right text version of the \affixline@num macro.

```

\ch@cksub@l@ckR 1232
\ch@ck@l@ckR 1233 \providecommand*{\l@dcalcnnum}[3]{%
\fx@l@cksR 1234 \ifnum #1 > #2\relax
\affixline@numR 1235 \l@dttempcnta = #1\relax
1236 \advance\@l@dttempcnta by -#2\relax
1237 \divide\@l@dttempcnta by #3\relax
1238 \multiply\@l@dttempcnta by #3\relax
1239 \advance\@l@dttempcnta by #2\relax
1240 \else
1241 \l@dttempcnta=#2\relax
1242 \fi}
1243
1244 \newcommand*{\ch@cksub@l@ckR}{%
1245 \ifcase\sub@lockR
1246 \or
1247 \ifnum\sublock@disp=\@ne
1248 \l@dttempcntb \z@ \l@dttempcnta \@ne
1249 \fi
1250 \or
1251 \ifnum\sublock@disp=\tw@
1252 \else
1253 \l@dttempcntb \z@ \l@dttempcnta \@ne
1254 \fi
1255 \or
1256 \ifnum\sublock@disp=\z@
1257 \l@dttempcntb \z@ \l@dttempcnta \@ne
1258 \fi
1259 \fi}
1260

```

```

1261 \newcommand*{\ch@ck@l@ckR}{%
1262   \ifcase\@lockR
1263   \or
1264     \ifnum\lock@disp=\@ne
1265       \@l@tempcntb \z@ \l@tempcnta \@ne
1266     \fi
1267   \or
1268     \ifnum\lock@disp=\tw@
1269     \else
1270       \@l@tempcntb \z@ \l@tempcnta \@ne
1271     \fi
1272   \or
1273     \ifnum\lock@disp=\z@
1274       \@l@tempcntb \z@ \l@tempcnta \@ne
1275     \fi
1276   \fi}
1277
1278 \newcommand*{\f@x@l@cksR}{%
1279   \ifcase\@lockR
1280   \or
1281     \global\@lockR \tw@
1282   \or \or
1283     \global\@lockR \z@
1284   \fi
1285   \ifcase\sub@lockR
1286   \or
1287     \global\sub@lockR \tw@
1288   \or \or
1289     \global\sub@lockR \z@
1290   \fi}
1291
1292
1293 \newcommand*{\affixline@numR}{%
1294   \ifl@dskipnotesR\else\ifnumberline
1295   \ifl@dskipnumber
1296     \global\l@dskipnumberfalse
1297   \else
1298     \ifsublines@
1299       \@l@tempcntb=\subline@numR
1300       \l@dcalcnm{\subline@numR}{\c@firstsublinenumR}{\c@sublinenumincrementR}%
1301       \ch@cksub@lockR
1302     \else
1303       \@l@tempcntb=\line@numR
1304       \ifx\linenumberlist\empty
1305         \l@dcalcnm{\line@numR}{\c@firstlinenumR}{\c@linenumincrementR}%
1306       \else
1307         \@l@tempcnta=\line@numR
1308         \edef\rem@inder{\linenumberlist,\number\line@numR,%
1309           \edef\sc@n@list{\def\noexpand\sc@n@list
1310             ###1,\number\@l@tempcnta,###2|{\def\noexpand\rem@inder{###2}}}%

```

```

1311 \sc@n@list\expandafter\sc@n@list\rem@inder|%
1312 \ifx\rem@inder\empty\advance\@l@tempcnta\@ne\fi
1313 \fi
1314 \ch@ck@l@ckR
1315 \fi
1316 \ifnum\@l@tempcnta=\@l@tempcntb
1317 \ifl@dskipversenumberR\else
1318 \if@twocolumn
1319 \if@firstcolumn
1320 \gdef\l@dld@ta{\llap{\leftlinenumR}}}%
1321 \else
1322 \gdef\l@drd@ta{\rlap{\rightlinenumR}}}%
1323 \fi
1324 \else
1325 \l@tempcntb=\line@marginR
1326 \ifnum\@l@tempcntb>\@ne
1327 \advance\@l@tempcntb by\page@numR
1328 \fi
1329 \ifodd\@l@tempcntb
1330 \gdef\l@drd@ta{\rlap{\rightlinenumR}}}%
1331 \else
1332 \gdef\l@dld@ta{\llap{\leftlinenumR}}}%
1333 \fi
1334 \fi
1335 \fi
1336 \fi
1337 \f@x@l@cksR
1338 \fi
1339 \fi
1340 \fi}

```

18.5 Pstart number printing in side

The printing of the pstart number is like in eledmac, 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 1341
\leftpstartnumR 1342 \newcommand*{\affixpstart@numL}{%
\rightpstartnumR 1343 \ifsidepstartnum
\leftpstartnumL 1344 \if@twocolumn
\rightpstartnumL 1345 \if@firstcolumn
\ifpstartnumR 1346 \gdef\l@dld@ta{\llap{\leftpstartnumL}}}%
1347 \else
1348 \gdef\l@drd@ta{\rlap{\rightpstartnumL}}}%

```

```

1349 \fi
1350 \else
1351 \l@dttempcntb=\line@margin
1352 \ifnum\l@dttempcntb>\@ne
1353 \advance\l@dttempcntb \page@num
1354 \fi
1355 \ifodd\l@dttempcntb
1356 \gdef\l@drd@ta{\rlap{{\rightpstartnumL}}}%
1357 \else
1358 \gdef\l@dld@ta{\llap{{\leftpstartnumL}}}%
1359 \fi
1360 \fi
1361 \fi
1362 }
1363 \newcommand*{\affixpstart@numR}{%
1364 \ifsidepstartnum
1365 \if@twocolumn
1366 \if@firstcolumn
1367 \gdef\l@dld@ta{\llap{{\leftpstartnumR}}}%
1368 \else
1369 \gdef\l@drd@ta{\rlap{{\rightpstartnumR}}}%
1370 \fi
1371 \else
1372 \l@dttempcntb=\line@marginR
1373 \ifnum\l@dttempcntb>\@ne
1374 \advance\l@dttempcntb \page@numR
1375 \fi
1376 \ifodd\l@dttempcntb
1377 \gdef\l@drd@ta{\rlap{{\rightpstartnumR}}}%
1378 \else
1379 \gdef\l@dld@ta{\llap{{\leftpstartnumR}}}%
1380 \fi
1381 \fi
1382 \fi
1383 }
1384
1385 \newcommand*{\leftpstartnumL}{%
1386 \ifpstartnum
1387 \thepstartL
1388 \kern\linenumsep\global\pstartnumfalse\fi
1389 }
1390 \newcommand*{\rightpstartnumL}{%
1391 \ifpstartnum\kern\linenumsep
1392 \thepstartL
1393 \global\pstartnumfalse\fi
1394 }
1395 \newif\ifpstartnumR
1396 \pstartnumRtrue
1397 \newcommand*{\leftpstartnumR}{%
1398 \ifpstartnumR

```



```

1399 \thepstartR
1400 \kern\linenumsep\global\pstartnumRfalse\fi
1401 }
1402 \newcommand*{\rightpstartnumR}{
1403 \ifpstartnumR\kern\linenumsep
1404 \thepstartR
1405 \global\pstartnumRfalse\fi
1406 }

```

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.

```
1407 \list@create{\inserts@listR}
```

`\add@insertsR` The right text version.

```

\add@inserts@nextR 1408 \newcommand*{\add@insertsR}{%
1409 \global\let\add@inserts@nextR=\relax
1410 \ifx\inserts@listR\empty \else
1411 \ifx\next@insertR\empty
1412 \ifx\insertlines@listR\empty
1413 \global\noteschanged@true
1414 \gdef\next@insertR{100000}%
1415 \else
1416 \gl@p\insertlines@listR\to\next@insertR
1417 \fi
1418 \fi
1419 \ifnum\next@insertR=\absline@numR
1420 \gl@p\inserts@listR\to\@insertR
1421 \@insertR
1422 \global\let\@insertR=\undefined
1423 \global\let\next@insertR=\empty
1424 \global\let\add@inserts@nextR=\add@insertsR
1425 \fi
1426 \fi
1427 \add@inserts@nextR}
1428

```

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@tempcnta` 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@tempcnta=\ballast@count
\ifnum\num@linesR>\@ne
\global\advance\par@lineR \@ne
\ifnum\par@lineR=\@ne
\advance\@l@tempcnta by \clubpenalty
\fi
\@l@tempcntb=\par@lineR \advance\@l@tempcntb \@ne
\ifnum\@l@tempcntb=\num@linesR
\advance\@l@tempcnta by \widowpenalty
\fi
\ifnum\par@lineR<\num@linesR
\advance\@l@tempcnta by \interlinepenalty
\fi
\fi
\ifnum\@l@tempcnta=\z@
\relax
\else
\ifnum\@l@tempcnta>-10000
\penalty\@l@tempcnta
\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.

```
1429 \newcommand*{\add@penaltiesL}{\}
1430 \newcommand*{\add@penaltiesR}{\}
1431
```

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.

```
1432 \newcommand*{\flush@notesR}{\%
1433 \xloop
1434 \ifx\inserts@listR\empty \else
1435 \glp\inserts@listR\to\@insertR
1436 \@insertR
1437 \global\let\@insertR=\undefined
1438 \repeat}
```

1439

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 21.3 p. 70 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

```

1440 \def\printlinesR#1|#2|#3|#4|#5|#6|#7|{\begingroup
1441   \setprintlines{#1}{#2}{#3}{#4}{#5}{#6}%
1442   \ifl@d@pnum #1\fullstop\fi
1443   \ifledplinenum \linenumr@p{#2}\Rlineflag\else \symlinenum\fi
1444   \ifl@d@ssub \fullstop \sublinenumr@p{#3}\fi
1445   \ifl@d@dash \endashchar\fi
1446   \ifl@d@pnum #4\fullstop\fi
1447   \ifl@d@elin \linenumr@p{#5}\Rlineflag\fi
1448   \ifl@d@esl \ifl@d@elin \fullstop\fi \sublinenumr@p{#6}\fi
1449 \endgroup}
1450
1451 \let\ledsavedprintlines\printlines
1452
```

19.2 Footnotes output specific to `\Pages`

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

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

- 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, 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`

```
1453 \newcommand\print@Xnotes@forpages[1]{%
```

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

```
1454 \ifcseempty{onlyXside@#1}{%
1455   \csuse{#1footstart}{#1}%
1456   \csuse{#1footgroup}{#1}%
1457 }%
```

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

```
1458 {%
1459   \ifboolexpr{%
1460     ((test {\ifcsstring{onlyXside@#1}{L}} and not test{\ifnumodd{\c@page}}))%
1461     or%
1462     (test {\ifcsstring{onlyXside@#1}{R}} and test{\ifnumodd{\c@page}}))%
1463   }%
```

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

```
1464   {%
1465     \correct@Xfootins@box{#1}%
1466     \csuse{#1footstart}{#1}%
1467     \csuse{#1footgroup}{#1}%
```

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

```
1468   \global\count\csuse{#1footins}=0%
1469   \global\skip\csuse{#1footins}=0pt%
```

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

```
1470   \csuse{Xnotefontsize@#1}%
1471   \global\advance\dimen\csuse{#1footins} by -\baselineskip%
```

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

```
1472   \global\boolfalse{keepforXside@#1}%
1473   }%
```

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.

```
1474   {%
1475     \global\booltrue{keepforXside@#1}%
```

Then restore expected rooms for notes on the next page.

```
1476   \global\count\csuse{#1footins}=\csuse{default@#1footins}%
1477   \global\skip\csuse{#1footins}=\csuse{beforeXnotes@#1}%
```

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

```
1478   \bgroup%
1479     \csuse{Xnotefontsize@#1}%
1480     \global\advance\dimen\csuse{#1footins} by \baselineskip%
1481   \egroup%
1482 % End of \cs{print@Xnotes@forpages}.
1483   }%
1484 }%
1485 }%
```

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

```
1486 \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.

```
1487 \ifbool{keepforXside@#1}{%
```

Some setting needed to do the right splitting.

```
1488 \csuse{Xnotefontsize@#1}%
```

```
1489 \splittopskip=0pt%
```

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

```
1490 \global\setbox\csuse{#1footins}=\vbox{%
```

```
1491 \vsplit\csuse{#1footins} to \dimexpr\ht\csuse{#1footins}-1pt\relax%
```

```
1492 \vskip \dimexpr-0.5\baselineskip-0.5\lineskip-0.5pt\relax%
```

```
1493 \unvbox\csuse{#1footins}%
```

```
1494 }%
```

End of the macro.

```
1495 }{}%
```

```
1496 }%
```

And now, the same for familiar footnotes.

```
1497 \newcommand\print@notesX@forpages[1]{%
```

```
1498 \ifcseempty{onlysideX@#1}{%
```

```
1499 \csuse{footstart#1}{#1}%
```

```
1500 \csuse{footgroup#1}{#1}%
```

```
1501 }%
```

```
1502 {%
```

```
1503 \ifboolexpr{%
```

```
1504 ((test {\ifcsstring{onlysideX@#1}{L}} and not test{\ifnumodd{\c@page}}))%
```

```
1505 or%
```

```
1506 (test {\ifcsstring{onlysideX@#1}{R}} and test{\ifnumodd{\c@page}}))%
```

```
1507 }%
```

```
1508 {%
```

```
1509 \correct@footinsX@box{#1}%
```

```
1510 \csuse{footstart#1}{#1}%
```

```
1511 \csuse{footgroup#1}{#1}%
```

```
1512 \global\count\csuse{footins#1}=0%
```

```
1513 \global\skip\csuse{footins#1}=0pt%
```

```
1514 \csuse{notefontsizeX@#1}%
```

```
1515 \global\advance\dimen\csuse{footins#1} by -\baselineskip%
```

```
1516 \global\boolfalse{keepforsideX@#1}%
```

```
1517 }%
```

```
1518 {%
```

```
1519 \global\booltrue{keepforsideX@#1}%
```

```
1520 \global\count\csuse{footins#1}=\csuse{default@footins#1}%
```

```
1521 \global\skip\csuse{footins#1}=\csuse{beforenotesX@#1}%
```

```
1522 \bgroup%
```

```
1523 \csuse{notefontsizeX@#1}%
```

```

1524         \global\advance\dimen\csuse{footins#1} by \baselineskip%
1525         \egroup%
1526     }%
1527 }%
1528 }%
1529 \newcommand{\correct@footinsX@box}[1]{%
1530     \ifbool{keepforsideX@#1}{%
1531         \csuse{notefontsizeX@#1}%
1532         \splittopskip=0pt%
1533         \global\setbox\csuse{footins#1}=\vbox{%
1534             \vsplit\csuse{footins#1} to \dimexpr\ht\csuse{footins#1}-1pt\relax%
1535             \vskip \dimexpr-0.5\baselineskip-0.5\lineskip-0.5pt\relax%
1536             \unvbox\csuse{footins#1}%
1537         }%
1538     }{}%
1539 }%

```

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.

```

1540 \list@create{\labelref@listR}
1541

```

`\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`.

```

1542 \def\l@dmake@labelsR#1|#2|#3|#4|#5{%
1543     \expandafter\ifx\csname the@label#5\endcsname \relax\else
1544         \led@warn@DuplicateLabel{#4}%
1545     \fi
1546     \expandafter\gdef\csname the@label#5\endcsname{#1|#2\Rlineflag|#3|#4}%
1547     \ignorespaces}
1548 \AtBeginDocument{%
1549     \def\l@dmake@labelsR#1|#2|#3|#4|#5{%
1550 }
1551

```

`\@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.

```

1552 \renewcommand*{\@lab}{%
1553     \ifledRcol
1554         \xright@appenditem{\linenumr@p{\line@numR}}{|%
1555         \ifsublines@ \sublinenumr@p{\subline@numR}\else 0\fi}%
1556     \to\labelref@listR

```

```

1557 \else
1558   \xright@appenditem{\linenumr@p{\line@num}}|{%
1559     \ifsublines@ \sublinenumr@p{\subline@num}\else 0\fi}%
1560   \to\labelref@list
1561 \fi}
1562

```

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* 1563 \WithSuffix\newcommand\sidenotemargin*[1]{%
1564   \l@dgetsidenote@margin{#1}
1565   \global\sidenote@marginR=\@l@dttempcntb
1566   \global\sidenote@margin=\@l@dttempcntb
1567 }
1568 \newcount\sidenote@marginR
1569 \global\sidenote@margin=\@ne
1570

```

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

```

1571 \newcommand*\affixside@noteR{%
1572   \def\sidenotecontent@{%}%
1573   \numgdef{\itemcount@}{0}%
1574   \def\do##1{%
1575     \ifnumequal{\itemcount@}{0}%
1576     {%
1577       \appto\sidenotecontent@{##1}}% Not print not separator before the 1st note
1578     {\appto\sidenotecontent@{\sidenotesep ##1}%
1579     }%
1580     \numgdef{\itemcount@}{\itemcount@+1}%
1581   }%
1582   \dolistloop{\l@dcnotes@text}%
1583   \ifnumgreater{\itemcount@}{1}{\led@err@ManySidenotes}{}%
1584   \gdef\@templ@d{%}%
1585   \gdef\@templ@n{\l@dcnotes@text\l@dcnotes@text@l\l@dcnotes@text@r}%
1586   \ifx\@templ@d\@templ@n \else%
1587     \if@twocolumn%
1588       \if@firstcolumn%
1589         \setl@dlp@rbox{##1}{\sidenotecontent@}%
1590       \else%
1591         \setl@drp@rbox{\sidenotecontent@}%
1592       \fi%
1593     \else%
1594       \l@dttempcntb=\sidenote@marginR%
1595       \ifnum\l@dttempcntb>\@ne%

```



```

1596     \advance\@l@tempcntb by\page@numR%
1597     \fi%
1598     \ifodd\@l@tempcntb%
1599         \setl@drp@rbox{\sidenotecontent@}%
1600         \gdef\sidenotecontent@{}%
1601         \numdef{\itemcount@}{0}%
1602         \dolistloop{\l@dcstotetext@l}%
1603         \ifnumgreater{\itemcount@}{1}{\led@err@ManyLeftnotes}{}%
1604         \setl@dlp@rbox{\sidenotecontent@}%
1605     \else%
1606         \setl@dlp@rbox{\sidenotecontent@}%
1607         \gdef\sidenotecontent@{}%
1608         \numdef{\itemcount@}{0}%
1609         \dolistloop{\l@dcstotetext@r}%
1610         \ifnumgreater{\itemcount@}{1}{\led@err@ManyRightnotes}{}%
1611         \setl@drp@rbox{\sidenotecontent@}%
1612     \fi%
1613 \fi%
1614 \fi%
1615 }
1616

```

22 Familiar footnotes

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

```

1617 \renewcommand{\l@dbfnote}[1]{%
1618     \ifnumberedpar@
1619     \gdef\@tag{#1\relax}%
1620     \ifledRcol%
1621         \xright@appenditem{\noexpand\vl@dbfnote{\expandonce\@tag}}{\@thefnmark}}%
1622         \to\inserts@listR
1623     \global\advance\insert@countR \@ne%
1624     \else%
1625         \xright@appenditem{\noexpand\vl@dbfnote{\expandonce\@tag}}{\@thefnmark}}%
1626         \to\inserts@list
1627     \global\advance\insert@count \@ne%
1628     \fi
1629 \fi\ignorespaces}
1630

```

`\normalbfnoteX`

```

1631 \renewcommand{\normalbfnoteX}[2]{%
1632     \ifnumberedpar@
1633     \ifledRcol%
1634     \ifluatex
1635         \footnotelang@lua[R]%
1636     \fi

```

```

1637 \ifundefined{xpg@main@language}%if polyglossia
1638 {}%
1639 {\footnotelang@poly[R]}%
1640 \protected@xdef\thisfootnote{\csuse{thefootnote#1}}%
1641 \xright@appenditem{\noexpand\vbfnoteX{#1}{#2}}{\expandonce\thisfootnote}}%
1642 \to\inserts@listR
1643 \global\advance\insert@countR \one%
1644 \else%
1645 \ifluatex
1646 \footnotelang@lua%
1647 \fi
1648 \ifundefined{xpg@main@language}%if polyglossia
1649 {}%
1650 {\footnotelang@poly}%
1651 \protected@xdef\thisfootnote{\csuse{thefootnote#1}}%
1652 \xright@appenditem{\noexpand\vbfnoteX{#1}{#2}}{\expandonce\thisfootnote}}%
1653 \to\inserts@list
1654 \global\advance\insert@count \one%
1655 \fi
1656 \fi\ignorespaces}
1657

```

23 Verse

Like in `eledmac`, the insertion of `hangingsymbol` is base on `\ifinserthangingsymbol`, and, for the right side, on `\ifinserthangingsymbolR`. Both commands also include the hanging space, to be sure the `\one@line` of hanging lines has the same width that the `\one@line` of normal lines and to prevent the column separator from shifting.

```

\inserthangingsymbolL
\inserthangingsymbolR 1658 \newif\ifinserthangingsymbolR
1659 \newcommand{\inserthangingsymbolL}{%
1660 \ifinserthangingsymbol%
1661 \ifinstanzaL%
1662 \hskip \ifundefined{sza@0@}{0}{\expandafter%
1663 \noexpand\csname sza@0@\endcsname}\stanzaindentbase%
1664 \hangingsymbol%
1665 \fi%
1666 \fi%
1667 }%
1668 \newcommand{\inserthangingsymbolR}{%
1669 \ifinserthangingsymbolR%
1670 \ifinstanzaR%
1671 \hskip \ifundefined{sza@0@}{0}{\expandafter%
1672 \noexpand\csname sza@0@\endcsname}\stanzaindentbase%
1673 \hangingsymbol%
1674 \fi%

```

```

1675 \fi%
1676 }%

```

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.

```

1677 \chardef\next=\catcode'\&
1678 \catcode'\&=\active
1679

```

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

```

1680 \newenvironment{astanza}{%
1681 \startstanzahook
1682 \catcode'\&=\active
1683 \global\stanza@count\@ne\stanza@modulo\@ne
1684 \ifnum\usernamecount{sza@00}=\z@
1685 \let\stanza@hang\relax
1686 \let\endlock\relax
1687 \else
1688 \rightskip\z@ plus 1fil\relax
1689 \fi
1690 \ifnum\usernamecount{szp@00}=\z@
1691 \let\sza@penalty\relax
1692 \fi
1693 \def&{%
1694 \endlock\mbox{}}%
1695 \sza@penalty
1696 \global\advance\stanza@count\@ne
1697 \@astanza@line}%
1698 \def\&{\@stopastanza}%
1699 \pstart
1700 \@astanza@line
1701 }{}
1702

```

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

```

1703 \newcommandx{\@stopastanza}[1][1,usedefault]{%
1704 \endlock\mbox{}}%
1705 \pend[#1]%
1706 \endstanzaextra%
1707 }%

```

`\@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.

```

1708 \newcommand*\@astanza@line{%
1709 \ifnum\value{stanzaindentrepetition}=0
1710 \parindent=\csname sza@number\stanza@count

```

```

1711             @\endcsname\stanzaindentbase
1712   \else
1713     \parindent=\csname sza@\number\stanza@modulo
1714             @\endcsname\stanzaindentbase
1715     \managestanza@modulo
1716   \fi
1717   \par
1718   \stanza@hang%\mbox{}%
1719   \ignorespaces}
1720

```

Lastly reset the modified category codes.

```

1721   \catcode'\&=\next
1722

```

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
\setnamebox  the regular box macros, but including the string ‘name’.
\unhnamebox 1723 \providecommand*\newnamebox}[1]{%
\unvnamebox 1724   \expandafter\newbox\csname #1\endcsname}
\namebox     1725 \providecommand*\setnamebox}[1]{%
               1726   \expandafter\setbox\csname #1\endcsname}
               1727 \providecommand*\unhnamebox}[1]{%
               1728   \expandafter\unhbox\csname #1\endcsname}
               1729 \providecommand*\unvnamebox}[1]{%
               1730   \expandafter\unvbox\csname #1\endcsname}
               1731 \providecommand*\namebox}[1]{%
               1732   \csname #1\endcsname}
               1733
\newnamecount  Macros for creating and using ‘named’ counts.
\usenamecount 1734 \providecommand*\newnamecount}[1]{%
               1735   \expandafter\newcount\csname #1\endcsname}
               1736 \providecommand*\usenamecount}[1]{%
               1737   \csname #1\endcsname}
               1738

```

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.

```
1739 \newcount\l@dc@maxchunks
1740 \newcommand{\maxchunks}[1]{\l@dc@maxchunks=#1}
1741 \maxchunks{5120}
1742
```

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

```
\l@dnumpstartsR 1743 \newcount\l@dnumpstartsR
1744
```

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

```
\l@pscR 1745 \newcount\l@dpscL
1746 \newcount\l@dpscR
1747
```

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

```
1748 \newcommand*\l@dsetuprawboxes{%
1749 \l@l@tempcntb=\l@dc@maxchunks
1750 \loop\ifnum\l@l@tempcntb>\z@
1751 \newnamebox{\l@dLcolrawbox\the\l@l@tempcntb}
1752 \newnamebox{\l@dRcolrawbox\the\l@l@tempcntb}
1753 \advance\l@l@tempcntb \m@ne
1754 \repeat}
1755
```

`\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.

```
1756 \newcommand*\l@dsetupmaxlinecounts{%
1757 \l@l@tempcntb=\l@dc@maxchunks
1758 \loop\ifnum\l@l@tempcntb>\z@
1759 \newnamecount{\l@dmaxlinesinpar\the\l@l@tempcntb}
1760 \advance\l@l@tempcntb \m@ne
1761 \repeat}
1762 \newcommand*\l@dzeromaxlinecounts{%
1763 \begingroup
1764 \l@l@tempcntb=\l@dc@maxchunks
1765 \loop\ifnum\l@l@tempcntb>\z@
1766 \global\usenamecount{\l@dmaxlinesinpar\the\l@l@tempcntb}=\z@
1767 \advance\l@l@tempcntb \m@ne
1768 \repeat
1769 \endgroup}
1770
```

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

```

1771 \AtBeginDocument{%
1772   \l@dssetuprawboxes
1773   \l@dssetupmaxlinecounts
1774   \l@dzetomaxlinecounts
1775   \l@dnumpstartsL=\z@
1776   \l@dnumpstartsR=\z@
1777   \l@dpscL=\z@
1778   \l@dpscR=\z@}
1779

```

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 1780 \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 \.
1781 \newcommand*{\l@dbbl@set@language}[1]{%
1782   \edef\language{#1}%
1783   \select@language{\language}%
1784   \if@files
1785     \protected@write\@auxout{}\string\select@language{\language}%
1786     \addtocontents{toc}{\string\select@language{\language}%
1787     \addtocontents{lof}{\string\select@language{\language}%
1788     \addtocontents{lot}{\string\select@language{\language}%
1789   \fi}
1790

```

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

```
\theledlanguageR 1791 \providecommand{\selectlanguage}[1]{%
1792 \newcommand*{\l@duselanguage}[1]{%
1793 \gdef\theledlanguageL{%
1794 \gdef\theledlanguageR{%
1795
```

Now do the babel fix or polyglossia, if necessary.

```
1796 \AtBeginDocument{%
1797 \ifundefined{xpg@main@language}{%
1798 \ifundefined{bbl@main@language}{%
```

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

```
1799 \l@dusedbabelfalse
1800 \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.

```
1801 \l@dusedbabeltrue
1802 \let\l@doldselectlanguage\selectlanguage
1803 \let\l@doldbbl@set@language\bbl@set@language
1804 \let\bbl@set@language\l@dbbl@set@language
1805 \renewcommand{\selectlanguage}[1]{%
1806 \l@doldselectlanguage{#1}%
1807 \ifledRcol \gdef\theledlanguageR{#1}%
1808 \else \gdef\theledlanguageL{#1}%
1809 \fi}
```

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

```
1810 \renewcommand*{\l@duselanguage}[1]{%
1811 \l@doldselectlanguage{#1}}
```

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

```
1812 \gdef\theledlanguageL{\bbl@main@language}%
1813 \gdef\theledlanguageR{\bbl@main@language}%
1814 }%
1815 }
```

If on Polyglossia

```
1816 { \let\old@otherlanguage\otherlanguage%
1817 \renewcommand{\otherlanguage}[2][]{%
1818 \selectlanguage[#1]{#2}%
1819 \ifledRcol \gdef\theledlanguageR{#2}%
1820 \else \gdef\theledlanguageL{#2}%
1821 \fi}%
1822 \let\l@duselanguage\select@language%
1823 \gdef\theledlanguageL{\xpg@main@language}%
1824 \gdef\theledlanguageR{\xpg@main@language}%
```

That's it.

```

1825 }}

\if@pstarts \check@pstarts returns \@pstartstrue if there are any unprocessed chunks.
\@pstartstrue 1826 \newif\if@pstarts
\@pstartsfalse 1827 \newcommand*{\check@pstarts}{%
\check@pstarts 1828 \@pstartsfalse
1829 \ifnum\l@dnumpstartsL>\l@dpscL
1830 \@pstartstrue
1831 \else
1832 \ifnum\l@dnumpstartsR>\l@dpscR
1833 \@pstartstrue
1834 \fi
1835 \fi
1836 }
1837

\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 1838 \newif\ifaraw@text
1839 \newcommand*{\checkraw@text}{%
1840 \araw@textfalse
1841 \ifvbox\namebox{l@dLcolrawbox\the\l@dpscL}
1842 \araw@texttrue
1843 \else
1844 \ifvbox\namebox{l@dRcolrawbox\the\l@dpscR}
1845 \araw@texttrue
1846 \fi
1847 \fi
1848 }
1849

\@writelinesinparL These write the number of text lines in a chunk to the section files, and then
\@writelinesinparR afterwards zero the counter.
1850 \newcommand*{\@writelinesinparL}{%
1851 \edef\next{%
1852 \write\linenum@out{\string\@pend[\the\@donereallinesL]}}%
1853 \next
1854 \global\@donereallinesL \z@}
1855 \newcommand*{\@writelinesinparR}{%
1856 \edef\next{%
1857 \write\linenum@outR{\string\@pendR[\the\@donereallinesR]}}%
1858 \next
1859 \global\@donereallinesR \z@}
1860

```


27 Parallel columns

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

```
\@eledsectionR 1861 \newsavebox{\@eledsectionL}%
                1862 \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.

```
1863 \newcommand*{\Columns}{%
1864   \l@dprintingcolumnstrue%
1865   \eledsection@correcting@skip=-\baselineskip% Correction for sections' titles
1866   \ifnum\l@dnumstartL=\l@dnumstartR\else
1867     \led@err@BadLeftRightPstarts{\the\l@dnumstartL}{\the\l@dnumstartR}%
1868   \fi
```

Start a group and zero counters, etc.

```
1869   \begingroup
1870     \l@dzeropenalties
1871     \endgraf\global\l@num@lines=\prevgraf
1872     \global\l@num@linesR=\prevgraf
1873     \global\l@par@line=\z@
1874     \global\l@par@lineR=\z@
1875     \global\l@dpscL=\z@
1876     \global\l@dpscR=\z@
```

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

```
1877   \check@pstarts
1878   \loop\if@pstarts
1879     \global\l@pstartnumtrue
1880     \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.

```
1881     \global\advance\l@dpscL \@ne
1882     \global\advance\l@dpscR \@ne
1883     \restore@pstartL@pc%
1884     \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

```
1885     \checkraw@text
1886   \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.

```
1887     \l@duselanguage{\the\l@languageL}%
1888     \do@lineL
1889     \xifinlist{\the\l@dpscL}{\eled@sections@{}}
```

```

1890      {%
1891      \ifdefstring{\@eledsectmark}{L}%
1892      {\csuse{eled@sectmark@the\l@dpscL}%
1893      }{}}%
1894      \global\csundef{eled@sectmark@the\l@dpscL}%
1895      \savebox{\@eledsectionL}{\parbox[t][t]{\Lcolwidth}{\vbox{}\print@eledse
1896      }%
1897      }{}%
1898      \l@duselanguage{\theledlanguageR}%
1899      \do@lineR
1900      \xifinlist{\the\l@dpscR}{\eled@sectionsR@@}
1901      {%
1902      \ifdefstring{\@eledsectmark}{R}%
1903      {\csuse{eled@sectmark@the\l@dpscR R}%
1904      }{}}%
1905      \global\csundef{eled@sectmark@the\l@dpscR R}%
1906      \savebox{\@eledsectionR}{\parbox[t][t]{\Rcolwidth}{\vbox{}\print@eledse
1907      }{}}%
1908      \hb@xt@ \hsize{%
1909      \ifdefstring{\columns@position}{L}{\}{\hfill }%
1910      \unhbox\l@dleftbox%
1911      \ifhbox\@eledsectionL%
1912      \usebox{\@eledsectionL}%
1913      \fi%
1914      \print@columnseparator%
1915      \unhbox\l@drightbox%
1916      \ifhbox\@eledsectionR%
1917      \usebox{\@eledsectionR}%
1918      \fi%
1919      \ifdefstring{\columns@position}{R}{\}{\hfill}}%
1920      }%
1921      \checkraw@text
1922      \checkverseL
1923      \checkverseR
1924      \checkpb@columns
1925      \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.

```

1926      \@writelinesinparL
1927      \@writelinesinparR
1928      \check@pstarts
1929      \ifbypstart@%
1930      \write\linenum@out{\string\@set[1]}
1931      \resetprevline@
1932      \fi
1933      \ifbypstart@R
1934      \write\linenum@outR{\string\@set[1]}
1935      \resetprevline@

```

```

1936      \fi
1937      \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.

```

1938      \flush@notes
1939      \flush@notesR
1940      \endgroup

1941      \global\l@dpscL=\z@
1942      \global\l@dpscR=\z@
1943      \global\l@dnumpstartsL=\z@
1944      \global\l@dnumpstartsR=\z@
1945      \l@dprintingcolumnsfalse%
1946      \ignorespaces
1947      \global\instanzaLfalse
1948      \global\instanzaRfalse}
1949

```

`\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).

```

1950 \def\print@columnseparator{%
1951   \ifdim\beforecolumnseparator<0pt%
1952     \hfill%
1953   \else%
1954     \hspace{\beforecolumnseparator}%
1955   \fi%
1956   \columnseparator%
1957   \ifdim\aftercolumnseparator<0pt%
1958     \hfill%
1959   \else%
1960     \hspace{\beforecolumnseparator}%
1961   \fi%
1962 }%
1963 %\end{macrocode}
1964 % \end{macro}
1965 % \begin{macro}{\checkpb@columns}
1966 % \cs{checkpb@columns} prevent or make pagebreaking in columns, depending of the use of \cs{ledpb} or
1967 % \begin{macrocode}
1968
1969 \newcommand{\checkpb@columns}{%
1970   \newif\if@pb
1971   \newif\if@nopb
1972   \IfStrEq{\led@pb@setting}{before}{
1973     \numdef{\next@absline}{\the\absline@num+1}%
1974     \numdef{\next@abslineR}{\the\absline@numR+1}%
1975     \xifinlistcs{\next@absline}{l@prev@pb}{\@pbtrue}{}%
1976     \xifinlistcs{\next@abslineR}{l@prev@pbR}{\@pbtrue}{%
1977     \xifinlistcs{\next@absline}{l@prev@nopb}{\@nopbtrue}{}%

```

```

1978 \xifinlistcs{\next@abslineR}{l@prev@nopbR}{\@nopbtrue}{%
1979 }{}
1980 \IfStrEq{\led@pb@setting}{after}{%
1981 \xifinlistcs{\the\absline@num}{l@prev@pb}{\@pbtrue}{}%
1982 \xifinlistcs{\the\absline@numR}{l@prev@pbR}{\@pbtrue}{%
1983 \xifinlistcs{\the\absline@num}{l@prev@nopb}{\@nopbtrue}{}%
1984 \xifinlistcs{\the\absline@numR}{l@prev@nopbR}{\@nopbtrue}{%
1985 }{}
1986 \if@nopb\nopagebreak[4]\enlargethispage{\baselineskip}\fi
1987 \if@pb\pagebreak[4]\fi
1988 }

```

`\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).

```

1989 \newcommand*{\columnseparator}{%
1990 \smash{\rule[-0.2\baselineskip]{\columnrulewidth}{1.05\baselineskip}}}
1991 \newdimen\columnrulewidth
1992 \columnrulewidth=\z@
1993

```

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

```

1994 \newcommand*{\columnspostion}[1]{%
1995 \xdef\columns@position{#1}%
1996 }%
1997 \xdef\columns@position{R}%

```

`\beforecolumnseparator` `\aftercolumnseparator` and `\beforecolumnseparator` 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`.

```

1998 \newlength{\beforecolumnseparator}%
1999 \setlength{\beforecolumnseparator}{-2pt}%
2000
2001 \newlength{\aftercolumnseparator}%
2002 \setlength{\aftercolumnseparator}{-2pt}%
2003

```

`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.

```

setwidthliketwocolumns@L
setpositionliketwocolumns@L
setnotepositionliketwocolumns@L
setwidthliketwocolumns@C
setpositionliketwocolumns@C
setnotepositionliketwocolumns@C
setwidthliketwocolumns@R
setpositionliketwocolumns@R
setnotepositionliketwocolumns@R

```

2004 `\newcommand{\setwidthliketwocolumns@L}{%`
2005 `% Temporary dimension, initially equal to the standard hsize, i.e. text width`

```

2006 % \begin{macrocode}
2007 \newdimen\temp%
2008 \temp=\hsize%

    Hsize : Left + Right width
2009 \hsize=\Lcolwidth%
2010 \advance\hsize\Rcolwidth%

    Now, calculating the remaining space
2011 \advance\temp-\hsize%

    And multiply the hsize by 2/3 of this space
2012 \multiply\temp by 2%
2013 \divide\temp by 3%
2014 \advance\hsize\temp%
2015 }%
2016
2017 \newcommand{\setpositionliketwocolumns@L}{%
2018 \renewcommand{\ledrlfill}{\hfill}%
2019 }%
2020
2021 \newcommand{\setnotespositionliketwocolumns@L}{%
2022 }%
2023
2024

2025 \newcommand{\setwidthliketwocolumns@C}{%
2026 % Temporary dimension, initially equal to the standard hsize, i.e. text width

2027 \newdimen\temp%
2028 \temp=\hsize%
2029 % Hsize : Left + Right width

2030 \hsize=\Lcolwidth%
2031 \advance\hsize\Rcolwidth%
2032 % Now, calculating the remaining space
2033 \advance\temp-\hsize%

    And multiply the hsize by 1/2 of this space
2034 \divide\temp by 2%
2035 \advance\hsize\temp%
2036 }%
2037
2038 \newcommand{\setpositionliketwocolumns@C}{%
2039 \doinsidelinehook{\hfill}%
2040 \renewcommand{\ledrlfill}{\hfill}%
2041 }%
2042
2043 \newcommand{\setnotespositionliketwocolumns@C}{%
2044 \newdimen\temp%
2045 \newdimen\tempa%
2046 \temp=\hsize%

```

```

2047 \tempa=\Lcolwidth%
2048 \advance\tempa\Rcolwidth%
2049 \advance\temp-\tempa%
2050 \divide\temp by 2%
2051 \leftskip=\temp%
2052 \rightskip=-\temp%
2053 }%
2054
2055 \newcommand{\setwidthliketwocolumns@R}{%
    Temporary dimension, initially equal to the standard hsize, i.e. text width
2056 \newdimen\temp%
2057 \temp=\hsize%
    Hsize : Left + Right width
2058 \hsize=\Lcolwidth%
2059 \advance\hsize\Rcolwidth%
    Now, calculating the remaining space
2060 \advance\temp-\hsize%
    And multiply the hsize by 2/3 of this space
2061 \multiply\temp by 2%
2062 \divide\temp by 3%
2063 \advance\hsize\temp%
2064 }%
2065
2066 \newcommand{\setpositionliketwocolumns@R}{%
2067 \doinsidelinehook{\hfill}%
2068 }%
2069
2070 \newcommand{\setnotespositionliketwocolumns@R}{%
2071 \newdimen\temp%
2072 \newdimen\tempa%
2073 \temp=\hsize%
2074 \tempa=\Lcolwidth%
2075 \advance\tempa\Rcolwidth%
2076 \advance\temp-\tempa%
2077 \divide\temp by 2%
2078 \leftskip=\temp%
2079 \rightskip=-\temp%
2080 }%
2081

```

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`

```

2082 \newcount\numpagelinesL
2083 \newcount\numpagelinesR
2084 \newcount\l@dminpagelines
2085

```

\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.

```

2086 \newcommand*{\Pages}{%
2087   \l@dprintingpagetrue%
2088   \eledsection@correcting@skip=-2\baselineskip% line correcting for section titles.
2089   \parledgroup@notespacing@set@correction%
2090   \typeout{}%
2091   \typeout{***** PAGES *****}%
2092   \ifnum\l@dnumstartsL=\l@dnumstartsR\else%
2093     \led@err@BadLeftRightPstarts{\the\l@dnumstartsL}{\the\l@dnumstartsR}%
2094   \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.

```

2095   \setlength{\Lcolwidth}{\textwidth}%
2096   \setlength{\Rcolwidth}{\textwidth}%

```

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

```

2097   \cleartol@devenpage%
2098   \begingroup%
2099     \l@dzeropenalties%
2100     \endgraf\global\num@lines=\prevgraf%
2101     \global\num@linesR=\prevgraf%
2102     \global\par@line=\z@%
2103     \global\par@lineR=\z@%
2104     \global\l@dpscL=\z@%
2105     \global\l@dpscR=\z@%
2106     \writtenlinesLfalse%
2107     \writtenlinesRfalse%

```

The footnotes are printed in a different way from expected in **eledmac**, as we may want to print the notes on one side only.

```

2108     \let\print@Xnotes\print@Xnotes@forpages%
2109     \let\print@notesX\print@notesX@forpages%

```

Check if there are chunks to be processed.

```

2110     \check@pstarts%
2111     \loop\if@pstarts%

```

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

```

2112       \global\advance\l@dpscL \@ne%
2113       \global\advance\l@dpscR \@ne%

```

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

```
2114 \getlinesfromparlistL%
2115 \getlinesfromparlistR%
2116 \l@dcalcmxoftwo{\@cs@linesinparL}{\@cs@linesinparR}%
2117 {\usernamecount{\l@dmxlinesinpar\the\l@dpscL}}%
2118 \check@pstarts%
2119 \repeat%
```

Zero the counts again, ready for the next bit.

```
2120 \global\l@dpscL=\z@%
2121 \global\l@dpscR=\z@%
```

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

```
2122 \getlinesfrompagelistL%
2123 \getlinesfrompagelistR%
2124 \l@dcalcmminoftwo{\@cs@linesonpageL}{\@cs@linesonpageR}%
2125 {\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.

```
2126 \check@pstarts%
2127 \if@pstarts%
```

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

```
2128 \global\advance\l@dpscL \@ne%
2129 \global\advance\l@dpscR \@ne%
2130 \restore@pstartL@pc%
2131 \restore@pstartR@pc%
```

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

```
2132 \global\@donereallinesL=\z@%
2133 \global\@donetotallinesL=\z@%
2134 \global\@donereallinesR=\z@%
2135 \global\@donetotallinesR=\z@%
```

Start a loop over the boxes (chunks).

```
2136 \checkraw@text%
2137 % \begingroup
2138 { \loop\ifaraw@text%
```

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

```
2139 \checkpageL%
2140 \l@duselanguage{\theledlanguageL}%
2141 { \loop\ifl@dsamepage%
```


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

```

2142         \ifdefstring{\@eledsectnotoc}{L}{\ledsectnotoc}{}%
2143         \csuse{before@pstartL@the\l@dpscL}%
2144         \global\csundef{before@pstartL@the\l@dpscL}%
2145         \do@lineL%
2146         \xifinlist{the\l@dpscL}{\eled@sections@@}
2147         {\print@eledsectionL}%
2148         {}%
2149         \advance\numpagelinesL \@one%

```

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`

```

2150         \ifshiftedpstarts%
2151             \ifdim\ht\l@dleftbox>0pt\hb@xt@%
2152                 \hsize{\ledstrutL\unhbox\l@dleftbox}%
2153             \else%
2154                 \dimen0=\pagetotal%
2155                 \advance\dimen0 by \baselineskip%
2156                 \global\pagetotal=\dimen0%
2157             \fi%
2158         \else%
2159             \parledgroup@correction@notespacing{L}
2160             \hb@xt@ \hsize{\ledstrutL\unhbox\l@dleftbox}%
2161         \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. Reset the verse skipnumber boolean

```

2162         \get@nextboxL%
2163         \global\l@dskipversenumberfalse%
2164         \ifprint@last@after@pendL%
2165             \csuse{after@pendL@the\l@dpscL}%
2166             \global\csundef{after@pendL@the\l@dpscL}%
2167         \fi%
2168         \checkpageL%
2169         \checkverseL%
2170         \checkpbL%
2171         \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.

```

2172      \ifl@dpagfull%
2173      \@writelinesonpageL{\the\numpagelinesL}%
2174      \else%
2175      \@writelinesonpageL{1000}%
2176      \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.

```

2177      \numpagelinesL \z%
2178      \parledgroup@correction@notespacing@init%
2179      \clearl@dleftpage }%

```

Now do the same for the right text.

```

2180      \checkpageR%
2181      \l@duselanguage{\theledlanguageR}%
2182 {
2183      \loop\ifl@dsamepage%
2184      \initnumbering@sectcountR%
2185      \ifdefstring{\@eledsectnotoc}{R}{\ledsectnotoc}{}%
2186      \csuse{before@pstartR@the\l@dpscR}%
2187      \global\csundef{before@pstartR@the\l@dpscR}%
2188      \do@lineR%
2189      \xifinlist{\the\l@dpscR}{\eled@sectionsR@@}%
2190      {\print@eledsectionR}%
2191      {}%
2192      \advance\numpagelinesR \one%
2193      \ifshiftedpstarts%
2194      \ifdim\ht\l@drightbox>0pt\hb@xt@%
2195      \hsize{\ledstrutR\unhbox\l@drightbox}%
2196      \else%
2197      \dimen0=\pagetotal%
2198      \advance\dimen0 by \baselineskip%
2199      \global\pagetotal=\dimen0%
2200      \fi%
2201      \else%
2202      \parledgroup@correction@notespacing{R}%
2203      \hb@xt@ \hsize{\ledstrutR\unhbox\l@drightbox}%
2204      \fi%
2205      \get@nextboxR%
2206      \global\l@dskipversenumberRfalse%
2207      \ifprint@last@after@pendR%
2208      \csuse{after@pendR@the\l@dpscR}%
2209      \global\csundef{after@pendR@the\l@dpscR}%
2210      \fi%
2211      \checkpageR%
2212      \checkverseR%
2213      \checkpbR%
2214      \repeat%

```

```

2214      \ifl@dpagfull%
2215      \@writelinesonpageR{\the\numpagelinesR}%
2216      \else%
2217      \@writelinesonpageR{1000}%
2218      \fi%
2219      \numpagelinesR=\z@%
2220      \parledgroup@correction@notespacing@init%

```

The page is full, so move onto the next (left, odd) page and repeat left text processing.

```
2221      \clearl@drighthpage}%
```

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.

```

2222      \checkraw@text%
2223      \ifaraw@text%
2224      \getlinesfrompagelistL%
2225      \getlinesfrompagelistR%
2226      \l@dcalc@minoftwo{\@cs@linesonpageL}{\@cs@linesonpageR}%
2227      {\l@dminpagelines}%
2228      \fi%
2229      \repeat}%

```

We have now output the text from all the chunks.

```
2230      \fi%
```

Make sure that there are no inserts hanging around.

```

2231      \flush@notes%
2232      \flush@notesR%
2233      \endgroup%

```

Zero counts ready for the next set of left/right text chunks. The boolean tests for stanza are switched to false.

```

2234      \global\l@dpscL=\z@%
2235      \global\l@dpscR=\z@%
2236      \global\l@dnumpstartsL=\z@%
2237      \global\l@dnumpstartsR=\z@%
2238      \global\instanzaLfalse%
2239      \global\instanzaRfalse%
2240      \l@dprintingpagesfalse%
2241      \finish@Pages@notes%Needed to prevent final notes overlap line number
2242      \ignorespaces}
2243

```

`\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.

```

2244 \newcommand{\finish@Pages@notes}{%
2245   \def\do##1{%
2246     \ifnocritical%
2247     \newbox\csuse{##1footins}
2248     \fi

```

```

2249 \ifnofamiliar@%
2250 \newbox\csuse{footins##1}
2251 \fi
2252 \ifvoid\csuse{##1footins}%
2253 \ifvoid\csuse{footins##1}\else%
2254 \newpage\null%
2255 \listbreak%
2256 \fi%
2257 \else%
2258 \newpage\null%
2259 \listbreak%
2260 \fi%
2261 }%
2262 \dolistloop{\@series}%
2263 }%

```

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

```

\ledstrutR 2264 \newcommand*\ledstrutL{\strut}
2265 \newcommand*\ledstrutR{\strut}
2266

```

`\cleartoevenpage` `\cleartoevenpage`, which is defined in the memoir class, is like `\clear(double)page`
`\cleartol@devenpage` 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.

```

2267 \providecommand{\cleartoevenpage}[1][\@empty]{%
2268 \clearpage
2269 \ifodd\c@page\hbox{##1}\clearpage\fi}
2270 \newcommand*\cleartol@devenpage{%
2271 \ifdim\pagetotal<\topskip% on an empty page
2272 \else
2273 \clearpage
2274 \fi
2275 \ifodd\c@page\hbox{}\clearpage\fi}

```

`\clearl@dleftpage` `\clearl@dleftpage` and `\clearl@drightpage` get us onto an odd and even page,
`\clearl@drightpage` 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`

```

2276 \newcommand*\clearl@dleftpage{%
2277 \ifdim\pagetotal=0pt\hbox{}\fi%
2278 \newpage%
2279 \ifodd\c@page\else
2280 \led@err@LeftOnRightPage
2281 \hbox{}\fi%
2282 \cleardoublepage
2283 \fi}
2284

```

```

2285 \newcommand*{\clearl@drighthpage}{%
2286   \ifdim\pagetotal=0pt\hbox{}\fi%
2287   \newpage%
2288   \ifodd\c@page
2289     \led@err@RightOnLeftPage
2290     \hbox{}\fi%
2291   \cleartoevenpage
2292   \fi}
2293

```

\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@linesinparL 2294 \newcommand*{\getlinesfromparlistL}{%
2295   \ifx\linesinpar@listL\empty
2296     \gdef\cs@linesinparL{0}%
2297   \else
2298     \gl@p\linesinpar@listL\to\cs@linesinparL
2299   \fi}
2300 \newcommand*{\getlinesfromparlistR}{%
2301   \ifx\linesinpar@listR\empty
2302     \gdef\cs@linesinparR{0}%
2303   \else
2304     \gl@p\linesinpar@listR\to\cs@linesinparR
2305   \fi}
2306

```

\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 to \getlinesfrompagelistR 1000. Similarly for \getlinesfrompagelistR.

```

\cs@linesonpageL 2307 \newcommand*{\getlinesfrompagelistL}{%
2308   \ifx\linesonpage@listL\empty
2309     \gdef\cs@linesonpageL{1000}%
2310   \else
2311     \gl@p\linesonpage@listL\to\cs@linesonpageL
2312   \fi}
2313 \newcommand*{\getlinesfrompagelistR}{%
2314   \ifx\linesonpage@listR\empty
2315     \gdef\cs@linesonpageR{1000}%
2316   \else
2317     \gl@p\linesonpage@listR\to\cs@linesonpageR
2318   \fi}
2319

```

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

```

2320 \newcommand*{\@writelinesonpageL}[1]{%
2321   \edef\next{\write\linenum@out{\string\@lopL{#1}}}%
2322   \next}

```

```

2323 \newcommand*{\@writelinesonpageR}[1]{%
2324   \edef\next{\write\linenum@outR{\string\@lopR{#1}}}%
2325   \next}
2326

```

`\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>`.

```

2327 \newcommand*{\l@dcalc@maxoftwo}[3]{%
2328   \ifnum #2>#1\relax
2329     #3=#2\relax
2330   \else
2331     #3=#1\relax
2332   \fi}
2333 \newcommand*{\l@dcalc@minoftwo}[3]{%
2334   \ifnum #2<#1\relax
2335     #3=#2\relax
2336   \else
2337     #3=#1\relax
2338   \fi}
2339

```

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

```

\checkpageL 2340 \newif\ifl@dsamepage
\checkpageR 2341 \l@dsamepagetrue
2342 \newif\ifl@dpagfull
2343
2344 \newcommand*{\checkpageL}{%
2345   \l@dpagfulltrue
2346   \l@dsamepagetrue
2347   \check@goal
2348   \ifdim\pagetotal<\ledthegoal
2349     \ifnum\numpagelinesL<\l@dminpagelines
2350     \else
2351       \l@dsamepagefalse
2352       \l@dpagfullfalse
2353     \fi
2354   \else
2355     \l@dsamepagefalse
2356     \l@dpagfulltrue
2357   \fi%
2358   \ifprint@last@after@pendL%
2359     \l@dpagfullfalse%

```

```

2360     \l@dsamepagefalse%
2361     \print@last@after@pendLfalse%
2362 \fi%
2363 }%
2364
2365 \newcommand*{\checkpageR}{%
2366     \l@dpagfulltrue
2367     \l@dsamepagetrue
2368     \check@goal
2369     \ifdim\pagetotal<\ledthegoal
2370         \ifnum\numpagelinesR<\l@dminpagelines
2371             \else
2372                 \l@dsamepagefalse
2373                 \l@dpagfullfalse
2374             \fi
2375         \else
2376             \l@dsamepagefalse
2377             \l@dpagfulltrue
2378         \fi%
2379     \ifprint@last@after@pendR%
2380         \l@dpagfullfalse%
2381         \l@dsamepagefalse%
2382     \print@last@after@pendRfalse%
2383 \fi%
2384 }%
2385

```

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

```

2386 \newcommand{\checkpbL}{
2387     \IfStrEq{\led@pb@setting}{after}{
2388         \xifinlistcs{\the\absline@num}{\l@prev@pb}{\l@dpagfulltrue\l@dsamepagefalse}{
2389             \xifinlistcs{\the\absline@num}{\l@prev@nopb}{\l@dpagfullfalse\l@dsamepagetrue}{
2390                 }{
2391                     \IfStrEq{\led@pb@setting}{before}{
2392                         \numdef{\next@absline}{\the\absline@num+1}
2393                         \xifinlistcs{\next@absline}{\l@prev@pb}{\l@dpagfulltrue\l@dsamepagefalse}{
2394                             \xifinlistcs{\next@absline}{\l@prev@nopb}{\l@dpagfullfalse\l@dsamepagetrue}{
2395                                 }{
2396                                     }
2397                     }
2398 \newcommand{\checkpbR}{
2399     \IfStrEq{\led@pb@setting}{after}{
2400         \xifinlistcs{\the\absline@numR}{\l@prev@pbR}{\l@dpagfulltrue\l@dsamepagefalse}{
2401             \xifinlistcs{\the\absline@numR}{\l@prev@nopbR}{\l@dpagfullfalse\l@dsamepagetrue}{
2402                 }{
2403                     \IfStrEq{\led@pb@setting}{before}{
2404                         \numdef{\next@abslineR}{\the\absline@numR+1}
2405                         \xifinlistcs{\next@abslineR}{\l@prev@pbR}{\l@dpagfulltrue\l@dsamepagefalse}{

```

```

2406     \xifinlistcs{\next@abslineR}{l@prev@nopbR}{\l@dpagfullfalse\l@dsamepagetrue}{}}
2407   }{}
2408 }

```

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

```

2409 \newcommand{\checkverseL}{
2410 \ifistanzaL
2411   \iflednopbinverse
2412     \ifinserthangingsymbol
2413       \numgdef{\prev@abslineverse}{\the\absline@num-1}
2414       \IfStrEq{\led@pb@setting}{after}{\lednopbnum{\prev@abslineverse}}{}
2415       \IfStrEq{\led@pb@setting}{before}{\ifnum\numpagelinesL<3\ledpbnum{\prev@abslineverse}}{}
2416     \fi
2417   \fi
2418 \fi
2419 }
2420 \newcommand{\checkverseR}{
2421 \ifistanzaR
2422   \iflednopbinverse
2423     \ifinserthangingsymbolR
2424       \numgdef{\prev@abslineverse}{\the\absline@numR-1}
2425       \IfStrEq{\led@pb@setting}{after}{\lednopbnumR{\prev@abslineverse}}{}
2426       \IfStrEq{\led@pb@setting}{before}{\ifnum\numpagelinesR<3\ledpbnumR{\prev@abslineverse}}{}
2427     \fi
2428   \fi
2429 \fi
2430 }

```

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

```

2431 \newdimen\ledthegoal
2432 \ifshiftedpstarts
2433   \newcommand*{\goalfraction}{0.95}
2434 \else
2435   \newcommand*{\goalfraction}{0.9}
2436 \fi
2437
2438 \newcommand*{\check@goal}{%
2439   \ledthegoal=\goalfraction\pagegoal}
2440

```

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

```

\ifwrittenlinesL 2441 \newif\ifwrittenlinesL
2442 \newif\ifwrittenlinesR
2443

```


`\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.

```

2444 \newcommand*{\get@nextboxL}{%
2445   \ifvbox\namebox{1@dLcolrawbox\the\1@dpscL}% box is not empty

   The current box is not empty; do nothing.
2446   \else%                               box is empty

   The box is empty. Check if enough lines (real and blank) have been output.
2447   \ifnum\usernamecount{1@dmaxlinesinpar\the\1@dpscL}>\@donetotallinesL
2448     \parledgroup@notes@endL
2449   \else

   Sufficient lines have been output.
2450     \ifnum\usernamecount{1@dmaxlinesinpar\the\1@dpscL}=\@donetotallinesL
2451       \parledgroup@notes@endL
2452     \fi
2453     \ifwrittenlinesL\else

   Write out the number of lines done, and set the boolean so this is only done once.
2454       \writelinesinparL
2455       \writtenlinesLtrue
2456     \fi
2457     \ifnum\1@dnumpestartsL>\1@dpscL

   There are still unprocessed boxes. Recalculate the maximum number of lines
   needed, and move onto the next box (by incrementing \1@dpscL). If needed, restart
   the line numbering.
2458       \writtenlinesLfalse
2459       \ifbypstart@
2460         \global\line@num=0%
2461         \resetprevline@%
2462       \fi
2463 % Add the content of the optional argument of the previous \cs{pend}.
2464 %   \begin{macrocode}
2465       \csuse{after@pendL@\the\1@dpscL}%
2466       \global\csundef{after@pendL@\the\1@dpscL}%

   Check the number of lines
2467       \1@dcalc@maxoftwo{\the\usernamecount{1@dmaxlinesinpar\the\1@dpscL}}%
2468                           {\the\@donetotallinesL}%
2469                           {\usernamecount{1@dmaxlinesinpar\the\1@dpscL}}%
2470       \global\@donetotallinesL \z@

   Go to the next pstart
2471       \global\advance\1@dpscL \@ne
2472       \global\pstartnumtrue%
2473       \restore@pstartL@pc%

   Add notes of parallel ledgroup.
2474       \parledgroup@notes@endL
2475       \parledgroup@correction@notes@spacing@final{L}
2476     \else

```

Add the content of the optional argument of the last `\pend`.

```

2477     \print@last@after@pendLtrue%
2478     \fi
2479     \fi
2480     \fi}

2481 \newcommand*{\get@nextboxR}{%
2482   \ifvbox\namebox{\l@dRcolrawbox\the\l@dpscR}% box is not empty
2483   \else% box is empty
2484     \ifnum\usernamecount{\l@dmaxlinesinpar\the\l@dpscR}>\@donetotallinesR
2485       \parledgroup@notes@endR
2486     \else
2487       \ifnum\usernamecount{\l@dmaxlinesinpar\the\l@dpscR}=\@donetotallinesR
2488         \parledgroup@notes@endR
2489       \fi
2490       \ifwrittenlinesR\else
2491         \@writelinesinparR
2492         \writtenlinesRtrue
2493       \fi
2494       \ifnum\l@dnumstartsR>\l@dpscR
2495         \writtenlinesRfalse
2496       \ifbypstartR
2497         \global\line@numR=0%
2498         \resetprevline%
2499       \fi
2500       \csuse{after@pendR@the\l@dpscR}%
2501       \global\csundef{after@pendR@the\l@dpscR}%
2502       \l@dcalc@maxoftwo{\the\usernamecount{\l@dmaxlinesinpar\the\l@dpscR}}%
2503         {\the\@donetotallinesR}%
2504         {\usernamecount{\l@dmaxlinesinpar\the\l@dpscR}}%
2505       \global\@donetotallinesR \z@
2506       \global\advance\l@dpscR \@ne
2507       \global\pstartnumRtrue%
2508       \restore@pstartR@pc%
2509       \parledgroup@notes@endR
2510       \parledgroup@correction@notes@spacing@final{R}
2511     \else
2512       \print@last@after@pendRtrue%
2513     \fi
2514   \fi
2515 \fi}
2516
```

29 Sections' titles' commands

As switching from left to right pages does not clear the page since v1.13.0, but only creates new pages, no `\vbox{}` is inserted, and consequently parallel chapters are mis-aligned.

So we patch the `\chapter` command in order to prevent this problem.

`\chapter`

```
2517 \pretocmd{\chapter}{%
2518   \ifl@printingpages%
2519     \vbox{}%
2520   \fi%
2521 }%
2522 {}%
2523 {}%
```

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

```
2524 \newcommand{\eledsectnotoc}[1]{\xdef\@eledsectnotoc{#1}}
2525 \eledsectnotoc{R}
```

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

```
2526 \newcommand{\eledsectmark}[1]{\xdef\@eledsectmark{#1}}
2527 \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`.

```
2528 \newskip\eledsection@correcting@skip
```

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

```
2529 \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`.

`\l@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).

```
2530 \def\l@prev@pbR{}
2531 \def\l@prev@nopbR{}
```

`\ledpbR` The `\ledpbR` macro writes the call to `\led@pbR` in line-list file. The `\ledpbnR` macro writes the call to `\led@pbnR` in line-list file. The `\lednopbR` macro writes the call to `\led@nopbR` in line-list file. The `\lednopbnR` macro writes the call to `\led@nopbnR` in line-list file.

```
2532 \newcommand{\ledpbR}{\write\linenum@outR{\string\led@pbR}}
2533 \newcommand{\ledpbnR}[1]{\write\linenum@outR{\string\led@pbnR{#1}}}
2534 \newcommand{\lednopbR}{\write\linenum@outR{\string\led@nopbR}}
2535 \newcommand{\lednopbnR}[1]{\write\linenum@outR{\string\led@nopbnR{#1}}}
```

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

```
2536 \newcommand{\led@pbR}{\listxadd{\l@prev@pbR}{\the\absline@numR}}
2537 \newcommand{\led@pbnumR}[1]{\listxadd{\l@prev@pbR}{#1}}
2538 \newcommand{\led@nopbR}{\listxadd{\l@prev@nopbR}{\the\absline@numR}}
2539 \newcommand{\led@nopbnumR}[1]{\listxadd{\l@prev@nopbR}{#1}}
```

31 Parallel ledgroup

`\parledgroup@` The marks `\parledgroup` contains information about the beginnings and endings
`\parledgroupseries@` of notes in a parallel ledgroup. `\parledgroupseries` contains the footnote series.
`\parledgroup@type@` `\parledgroupseries` contains the type of the footnote: critical (Xfootnote) or
familiar (footnoteX).

```
2540 \newmarks\parledgroup@
2541 \newmarks\parledgroup@series
2542 \newmarks\parledgroup@type
```

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

```
2543 \newcommand{\parledgroup@notes@startL}{%
2544   \ifnum\usenamecount{\l@maxlinesinpar\the\l@dpscL}>0%
2545     \IfStrEq{\splitfirstmarks\parledgroup@type}{footnoteX}{\csuse{bhooknoteX@\splitfirstmarks}}
2546     \IfStrEq{\splitfirstmarks\parledgroup@type}{Xfootnote}{\csuse{bhookXnote@\splitfirstmarks}}
2547   \fi%
2548   \global\ledgroupnotesL@true%
2549   \insert@noterule@ledgroup{L}%
2550 }
2551 \newcommand{\parledgroup@notes@startR}{%
2552   \ifnum\usenamecount{\l@maxlinesinpar\the\l@dpscR}>0%
2553     \IfStrEq{\splitfirstmarks\parledgroup@type}{footnoteX}{\csuse{bhooknoteX@\splitfirstmarks}}
2554     \IfStrEq{\splitfirstmarks\parledgroup@type}{Xfootnote}{\csuse{bhookXnote@\splitfirstmarks}}
2555   \fi%
2556   \global\ledgroupnotesR@true%
2557   \insert@noterule@ledgroup{R}%
2558 }
```

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

```
2559 \newcommand{\parledgroup@notes@endL}{%
2560   \global\ledgroupnotesL@false%
2561 }
2562 \newcommand{\parledgroup@notes@endR}{%
2563   \global\ledgroupnotesR@false%
2564 }
```

`\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`

```

2565 \newcommand{\insert@noterule@ledgroup}[1]{
2566   \IfStrEq{\splitbotmarks\parledgroup@}{begin}{%
2567     \IfStrEq{\splitbotmarks\parledgroup@type}{Xfootnote}{
2568       \csuse{ifledgroupnotes#1@}
2569       \vskip\skip\csuse{mp\splitbotmarks\parledgroup@series footins}
2570       \csuse{\splitbotmarks\parledgroup@series footnoterule}
2571       \fi
2572     }
2573   {}
2574   \IfStrEq{\splitbotmarks\parledgroup@type}{footnoteX}{
2575     \csuse{ifledgroupnotes#1@}
2576     \vskip\skip\csuse{mpfootins\splitbotmarks\parledgroup@series}
2577     \csuse{footnoterule\splitbotmarks\parledgroup@series}
2578     \fi
2579   }{}
2580 }
2581 {}
2582 }

```

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

```

2583 \newcommand{\parledgroupnotespacing}{}

```

`up@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`.

```

2584 \dimdef{\parledgroup@notespacing@correction}{0pt}
2585 \newcommand{\parledgroup@notespacing@set@correction}{%
2586   {\notefontsetup\parledgroupnotespacing\dimgdef{\temp@spacing}{\baselineskip}}%
2587   \dimgdef{\parledgroup@notespacing@correction}{\baselineskip-\temp@spacing}%
2588 }

```

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

```

2589 \newcommand{\parledgroup@correction@notespacing@init}{
2590   \dimdef{\parledgroup@notespacing@correction@accumulated}{0pt}
2591   \dimdef{\parledgroup@notespacing@correction@modulo}{0pt}
2592 }
2593 \parledgroup@correction@notespacing@init

```

`rrrection@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.

```

2594 \newcommand{\parledgroup@correction@notespacing@final}[1]{
2595     \ifparledgroup
2596     \vspace{\parledgroup@notespacing@correction@accumulated}
2597     \parledgroup@correction@notespacing@init%
2598     \ifstrequal{#1}{L}{
2599         \numdef{\@checking}{\the\l@dpscL-1}
2600     }{
2601         \numdef{\@checking}{\the\l@dpscR-1}
2602     }
2603     \dimdef{\@beforenotes@current@diff}{\csuse{\parledgroup@beforenotes@\@checking L}-\cs
2604     \ifstrequal{#1}{L}%
2605         {% Left
2606         \ifdimgreater{\@beforenotes@current@diff}{0pt}{\vspace{-\@beforenotes@current@diff}
2607         }%
2608         {% Right
2609         \ifdimgreater{\@beforenotes@current@diff}{0pt}{\vspace{\@beforenotes@current@diff}}
2610         }%
2611     \fi
2612 }

```

`\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.

```

2613 {}
2614 \newcommand{\parledgroup@correction@notespacing}[1]{%
2615     \csuse{ifledgroupnotes#1@}%
2616     \vspace{-\parledgroup@notespacing@correction}%
2617     \dimdef{\parledgroup@notespacing@correction@accumulated}{\parledgroup@notespacing@correction}
2618     \dimdef{\parledgroup@notespacing@correction@modulo}{\parledgroup@notespacing@correction}
2619     \ifdimless{\parledgroup@notespacing@correction@modulo}{\baselineskip}{\advance\parledgroup@notespacing@correction@modulo\baselineskip}
2620     \dimdef{\parledgroup@notespacing@correction@modulo}{\parledgroup@notespacing@correction}
2621     }% mean greater than equal
2622     \fi%
2623 }

```

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

```

2624 \dimdef\parledgroup@beforenotesL{0pt}
2625 \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.

```

2626 \newcommand{\parledgroup@beforenotes@save}[1]{
2627   \ifparledgroup
2628     \csdimgdef{@parledgroup@beforenotes@the\csuse{1@dnumstarts#1}#1}{\csuse{parledgroup@beforenotes
2629     \csdimgdef{parledgroup@beforenotes#1}{Opt}
2630   \fi
2631 }
```

32 The End

`i/codei`

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>	370, 621, 622
<code>\&</code>	1677, 1678, 1682, 1698, 1721	<code>\@afterindentfalse</code> 769
<code>\@RTLtrue</code>	1045	<code>\@arabic</code> 222, 223, 817, 819

- \@astanza@line 1697, 1700, 1708
 - \@auxout 1785
 - \@beforenotes@current@diff 2603, 2606, 2609
 - \@chapter 770
 - \@checking 2599, 2601, 2603
 - \@cs@linesinparL 2116, 2294
 - \@cs@linesinparR 2116, 2294
 - \@cs@linesonpageL 2124, 2226, 2307
 - \@cs@linesonpageR 2124, 2226, 2307
 - \@currentlabel 863, 911
 - \@donereallinesL 975, 1010, 1852, 1854, 2132
 - \@donereallinesR 975, 1103, 1857, 1859, 2134
 - \@donetotallinesL 975, 1011, 1014, 2133, 2447, 2450, 2468, 2470
 - \@donetotallinesR 975, 1104, 1107, 2135, 2484, 2487, 2503, 2505
 - \@eled@sectioningfalse 1049
 - \@eled@sectioningtrue 1039
 - \@eledsectionL 1861, 1895, 1911, 1912
 - \@eledsectionR 1861, 1906, 1916, 1917
 - \@eledsectmark 1036, 1891, 1902, 2526
 - \@eledsectnotoc 1035, 2142, 2184, 2524
 - \@gobble 550
 - \@gobbletwo 555
 - \@insertR 1420–1422, 1435–1437
 - \@l@dttempcnta 443, 445, 447, 448, 452, 454, 456, 457, 1165, 1209, 1210, 1212, 1214, 1217, 1218, 1235–1239, 1241, 1248, 1253, 1257, 1265, 1270, 1274, 1307, 1310, 1312, 1316
 - \@l@dttempcntb 178, 180, 182, 1200, 1201, 1248, 1253, 1257, 1265, 1270, 1274, 1299, 1303, 1316, 1325–1327, 1329, 1351–1353, 1355, 1372–1374, 1376, 1565, 1566, 1594–1596, 1598, 1749–1753, 1757–1760, 1764–1767
 - \@lab 554, 1552
 - \@lock 993, 1143
 - \@lockR 49, 314, 316, 318, 331, 477, 493, 494, 496, 497, 525, 526, 528, 1085, 1124, 1171, 1173, 1174, 1176, 1262, 1279, 1281, 1283
 - \@lopL 579, 2321
 - \@lopR 550, 579, 2324
 - \@nl 292, 604, 606
 - \@nl@reg 341
 - \@nl@regR 292
 - \@nobreakfalse 825, 874, 1038
 - \@nobreaktrue 823, 827, 872, 876, 1038
 - \@noprtrue 1977, 1978, 1983, 1984
 - \@oldnbreak 823, 825, 872, 874, 930, 952
 - \@pbtrue 1975, 1976, 1981, 1982
 - \@pend 570, 1852
 - \@pendR 570, 1857
 - \@pstartfalse 1826
 - \@pstartstrue 1826
 - \@ref 540
 - \@ref@reg 568
 - \@schapter 770
 - \@series 715, 2262
 - \@set 402, 628, 629, 1930, 1934
 - \@startstanza 782, 783, 805, 806
 - \@stopastanza 1698, 1703
 - \@sw 555
 - \@tag 1619, 1621, 1625
 - \@temp 732, 733, 738, 739
 - \@templ@d 1584, 1586
 - \@templ@n 1585, 1586
 - \@tmp 687, 689, 690, 693–695
 - \@tmpa 688, 689, 694, 697
 - \@writelinesinparL 1850, 1926, 2454
 - \@writelinesinparR 1850, 1927, 2491
 - \@writelinesonpageL 2173, 2175, 2320
 - \@writelinesonpageR 2215, 2217, 2320
 - \@xloop 1433
- A**
- \absline@num 366, 436, 450, 469, 1133, 1973, 1981, 1983, 2388, 2389, 2392, 2413
 - \absline@numR 44, 239, 294, 297, 300, 433, 441, 462, 481, 515, 545, 1114, 1155, 1156, 1200, 1419, 1974, 1982, 1984, 2400, 2401, 2404, 2424, 2536, 2538
 - \actionlines@list 284, 287, 436, 450, 469
 - \actionlines@listR 243, 260, 276, 279, 433, 441, 462, 481, 515, 1222, 1225
 - \actions@list 288, 437, 457, 471, 473
 - \actions@listR 243, 261, 280, 434, 448, 464, 466, 483, 492, 517, 524, 1226
 - \add@inserts 1007, 1021

- \add@inserts@nextR 1408
 - \add@insertsR 1099, 1408
 - \add@penaltiesL 1009, 1429
 - \add@penaltiesR 1102, 1429
 - \addtocontents 1786–1788
 - \addtocounter 933, 955, 1034
 - \advanceline 620
 - \affixline@num 1004
 - \affixline@numR 1096, 1232
 - \affixpstart@numL 1019, 1341
 - \affixpstart@numR 1341
 - \affixside@note 1007, 1021
 - \affixside@noteR 1099, 1571
 - \aftercolumnseparator . . . 4, 1957, 1998
 - \appto 1577, 1578
 - \araw@textfalse 1838
 - \araw@texttrue 1838
 - astanza (environment) 10, 1680
 - \at@begin@pairs 746, 751, 752
 - \at@every@pend 937, 959
 - \at@every@pstart 866, 914
 - \at@every@pstart@call 868, 916, 963, 964
 - \AtBeginDocument . . . 1548, 1771, 1796
 - \AtBeginPairs 3, 751
 - \AtEveryPstartCall 963
- B**
- \ballast@count 1153, 1158
 - \bbl@main@language 1812, 1813
 - \bbl@set@language 1803, 1804
 - \beforecolumnseparator 4, 1951, 1954, 1960, 1998
 - \beginnumbering 9, 793, 835
 - \beginnumberingR . . . 35, 116, 793, 884
 - \bfseries 817, 819
 - \boolfalse 1472, 1516
 - \booltrue 1475, 1519
 - \bypage@Rfalse 136, 152, 160
 - \bypage@Rtrue 136, 144
 - \bypstart@Rfalse 136, 145, 161
 - \bypstart@Rtrue 136, 153
- C**
- \c@ballast 1158
 - \c@chapter 97
 - \c@chapterR 97
 - \c@firstlinenumR 187, 1305
 - \c@firstsublinenumR 191, 1300
 - \c@linenumincrementR 187, 1305
 - \c@page 604, 606, 1460, 1462, 1504, 1506, 2269, 2275, 2279, 2288
 - \c@pstartL 721, 733, 817
 - \c@pstartR 726, 739, 819
 - \c@section 98
 - \c@sectionR 98
 - \c@sublinenumincrementR . . 191, 1300
 - \c@subsection 99
 - \c@subsectionR 99
 - \c@subsubsection 100
 - \c@subsubsectionR 100
 - \ch@ck@l@ckR 1232
 - \ch@cksub@l@ckR 1232
 - \ch@cksub@lockR 1301
 - \chapter 755, 756, 765, 2517
 - \chapterinpages 742, 756, 767
 - \chardef 1677
 - \check@goal 2347, 2368, 2431
 - \check@pstarts 1826, 1877, 1928, 2110, 2118, 2126
 - \checkpageL 2139, 2168, 2340
 - \checkpageR 2180, 2210, 2340
 - \checkpb@columns . . . 1924, 1965, 1969
 - \checkpbL 2170, 2386
 - \checkpbR 2212, 2386
 - \checkraw@text 1838, 1885, 1921, 2136, 2222
 - \checkverseL 1922, 2169, 2409
 - \checkverseR 1923, 2211, 2409
 - \cleardoublepage 2282
 - \clearl@dleftpage 2179, 2276
 - \clearl@drighthpage 2221, 2276
 - \cleartoevenpage 2267, 2291
 - \cleartol@devenpage 2097, 2267
 - \closeout 88, 592, 599
 - \columnrulewidth 4, 1989
 - \Columns 3, 1863
 - \columns@position . . . 1909, 1919, 1994
 - \columnseparator 4, 1956, 1989
 - \columnspan 4, 1994
 - \correct@footinsX@box 1453
 - \correct@Xfootins@box 1453
 - \count 1468, 1476, 1512, 1520
 - \countLline 970, 982
 - \countRline 970, 1074
 - \critext 649
 - \cs 1482, 1966, 2463
 - \csdingdef 2628, 2629
 - \csgdef 672, 677, 866, 867, 914, 915, 937, 938, 959, 960

- `\cslet` 690, 695, 722, 727
`\csundef` 1050, 1894, 1905,
 2144, 2166, 2186, 2208, 2466, 2501
`\csuse` 685, 688, 698,
 699, 1047, 1455, 1456, 1466–
 1471, 1476, 1477, 1479, 1480,
 1488, 1490, 1491, 1493, 1499,
 1500, 1510–1515, 1520, 1521,
 1523, 1524, 1531, 1533, 1534,
 1536, 1640, 1651, 1892, 1903,
 2143, 2165, 2185, 2207, 2247,
 2250, 2252, 2253, 2465, 2500,
 2545, 2546, 2553, 2554, 2568–
 2570, 2575–2577, 2603, 2615, 2628
- ### D
- `\DeclareOption` 9–12
`\def@tempb` 158
`\dimdef` 2584, 2590, 2591,
 2603, 2617, 2618, 2620, 2624, 2625
`\dimen` 607,
 608, 612–614, 618, 1471, 1480,
 1515, 1524, 2154–2156, 2196–2198
`\dimexpr` 1491, 1492, 1534, 1535
`\dingdef` 2586, 2587
`\divide` 1237, 2013, 2034, 2050, 2062, 2077
`\do@actions` 1134
`\do@actions@fixedcodeR` 1162
`\do@actions@nextR` 1162
`\do@actionsR` 1115, 1162
`\do@ballast` 1135
`\do@ballastR` 1116, 1153
`\do@insidelineLhook` . 1024, 1063, 1066
`\do@insidelineRhook` 1064, 1066
`\do@lineL` 980, 1888, 2145
`\do@lineLhook` 986, 1061, 1066
`\do@lineR` 1071, 1899, 2187
`\do@lineRhook` 1062, 1066, 1078
`\do@lockoff` 512
`\do@lockoffL` 536
`\do@lockoffR` 512
`\do@lockon` 477
`\do@lockonL` 509
`\do@lockonR` 477
`\doinsidelinehook` 2039, 2067
`\doinsidelineLhook` 1061
`\doinsidelineRhook` 1061
`\dolineLhook` 1061
`\dolineRhook` 1061
`\dolistloop` 715, 1582, 1602, 1609, 2262
- `\dummy@ref` 549
`\dump@pstartL@pc` 720, 931
`\dump@pstartR@pc` 720, 953
- ### E
- `\edfont@info` 653, 656, 662, 665
`\edlabel` 1542
`\edtext` 649
`\eled@sectioningR@out` .. 63, 88, 2529
`\eled@sections@@` 1006, 1038, 1889, 2146
`\eled@sectionsR@@` 60, 1098, 1900, 2188
`\eledpar@error` 19, 21, 24, 27, 29
`\eledsection@correcting@skip` ...
 1058, 1865, 2088, 2528
`\eledsectmark` 14, 2526
`\eledsectnotoc` 14, 2524
`\empty` 73, 76, 276, 284,
 651, 660, 731, 737, 843, 892,
 1222, 1304, 1312, 1410–1412,
 1423, 1434, 2295, 2301, 2308, 2314
`\endashchar` 1445
`\endgraf` 926, 948, 1871, 2100
`\endline@num` 558, 564
`\endlock` 640, 1686, 1694, 1704
`\endnumbering` 9, 66, 120, 794
`\endnumberingR` 38, 66, 103, 115, 128, 794
`\endpage@num` 557, 564
`\endstanzaextra` 1706
`\endsub` 607
`\endsubline@num` 559, 565
`\enlargethispage` 1986
 environments:
 `astanza` 10, 1680
 `Leftside` 7, 774
 `pages` 5, 742
 `pairs` 3, 742
 `Rightside` 7, 791
`\expandonce` 1621, 1625, 1641, 1652
`\extensionchars` 55, 109, 125, 133
- ### F
- `\f@x@l@cks` 1002
`\f@x@l@cksR` 1094, 1232
`\finish@Pages@notes` 2241, 2244
`\first@linenum@out@Rfalse` .. 587, 593
`\first@linenum@out@Rtrue` 587
`\firstlinenum` 7, 196
`\firstlinenum*` 7, 196
`\firstsublinenum` 7, 196
`\firstsublinenum*` 7, 196

\fix@page	337, <u>344</u>	\ifbypage@	360
\flag@end	<u>607</u>	\ifbypage@R	<u>136</u> , 350, 1204
\flag@start	<u>607</u>	\ifbypstart@	572, 1929, 2459
\flush@notes	1938, 2231	\ifbypstart@R	<u>136</u> , 576, 1933, 2496
\flush@notesR	<u>1432</u> , 1939, 2232	\ifcseempty	1454, 1498
\footnotelang@lua	1635, 1646	\ifcsstring	1460, 1462, 1504, 1506
\footnotelang@poly	1639, 1650	\ifdefstring	1035, 1036, 1045, 1891, 1902, 1909, 1919, 2142, 2184
\footnotexmk	6	\ifdim	608, 612, 614, 618, 1951, 1957, 2151, 2193, 2271, 2277, 2286, 2348, 2369
\footnotexnomk	6	\ifdimgreater	2606, 2609
\fullstop	235, 1442, 1444, 1446, 1448	\ifdimless	2619
G			
\get@linelistfile	272	\iffirst@linenum@out@R	<u>587</u> , 591
\get@nextboxL	2162, <u>2444</u>	\ifhbox	1911, 1916
\get@nextboxR	2204, <u>2444</u>	\ifinserthangingsymbol	1660, 2412
\getline@numL	992, 1132	\ifinserthangingsymbolR	1658, 1669, 2423
\getline@numR	1084, <u>1113</u>	\ifinstanzaL	<u>772</u> , <u>772</u> , 1661, 2410
\getlinesfrompagelistL	2122, 2224, <u>2307</u>	\ifinstanzaR	<u>772</u> , 773, 1670, 2421
\getlinesfrompagelistR	2123, 2225, <u>2307</u>	\ifl@d@dash	1445
\getlinesfromparlistL	2114, <u>2294</u>	\ifl@d@elin	1447, 1448
\getlinesfromparlistR	2115, <u>2294</u>	\ifl@d@esl	1448
\gl@p	279, 280, 287, 288, 655, 664, 694, 732, 738, 1225, 1226, 1416, 1420, 1435, 2298, 2304, 2311, 2317	\ifl@d@pnum	1442, 1446
\goalfraction	5, <u>2431</u>	\ifl@d@ssub	1444
H			
\hangingsymbol	12, 1664, 1673	\ifl@dhiddenumber	1000, 1092
\hb@xt@	999, 1013, 1023, 1091, 1106, 1908, 2151, 2160, 2193, 2202	\ifl@dpagefull	2172, 2214, <u>2340</u>
\hsize	861, 909, 1908, 2008– 2011, 2014, 2028, 2030, 2031, 2033, 2035, 2046, 2057–2060, 2063, 2073, 2152, 2160, 2194, 2202	\ifl@dpadding	<u>14</u>
I			
\if@filesw	1784	\ifl@dpairing	<u>14</u> , 70
\if@firstcolumn	1319, 1345, 1366, 1588	\ifl@dprintingpages	2518
\if@nobreak	822, 871	\ifl@dssamelang	<u>1781</u>
\if@noeled@sec	61, 87	\ifl@dsamepage	2141, 2182, <u>2340</u>
\if@nopb	1971, 1986	\ifl@dskipnumber	1295
\if@pb	1970, 1987	\ifl@dskipversenumberR	1163, 1317
\if@pstarts	<u>1826</u> , 1878, 2111, 2127	\ifl@dusedbabel	<u>1780</u>
\if@RTL	1051	\iflabelpstart	863, 911
\ifaraw@text	<u>1838</u> , 1886, 2138, 2223	\ifledgroupnotesL@	1136
\ifautopar	853, 902	\ifledgroupnotesR@	1117, 1294
\ifbool	1487, 1530	\iflednopbinverse	2411, 2422
\ifboolexpr	1459, 1503	\ifledplinenum	1443
		\ifledRcol	<u>14</u> , 179, 201, 205, 209, 213, 257, 274, 338, 347, 372, 386, 403, 420, 432, 440, 461, 506, 533, 542, 609, 615, 621, 628, 636, 641, 645, 650, 1553, 1620, 1633, 1807, 1819
		\ifluatex	828, 877, 1025, 1041, 1634, 1645
		\ifnocritical@	146, 154, 162, 671, 705, 2246

- \ifnofamiliar@ ... 675, 679, 708, 2249
 - \ifnoteschanged@ 80
 - \ifnumberedpar@
 - ... 837, 886, 922, 944, 1618, 1632
 - \ifnumbering 139, 833, 919
 - \ifnumberingR ... 36, 67, 105, 882, 941
 - \ifnumberline 1118, 1137, 1294
 - \ifnumberpstart 722,
 - 727, 854, 903, 932, 954, 981, 1072
 - \ifnumequal 1575
 - \ifnumgreater 1583, 1603, 1610
 - \ifnumodd 1460, 1462, 1504, 1506
 - \ifodd 1329, 1355,
 - 1376, 1598, 2269, 2275, 2279, 2288
 - \ifparledgroup 2595, 2627
 - \ifprint@last@after@pendL
 - 965, 2164, 2358
 - \ifprint@last@after@pendR
 - 965, 2206, 2379
 - \ifpst@rtedL 32, 841
 - \ifpst@rtedR 32, 890
 - \ifpstartnum 1386, 1391
 - \ifpstartnumR 1341
 - \ifshiftedpstarts 6, 2150, 2192, 2432
 - \ifsidepstartnum 855, 904, 1343, 1364
 - \ifstreempty 865, 913, 936, 958
 - \IfStrEq 990, 1082, 1972,
 - 1980, 2387, 2391, 2399, 2403,
 - 2414, 2415, 2425, 2426, 2545,
 - 2546, 2553, 2554, 2566, 2567, 2574
 - \ifstrequal 2598, 2604
 - \ifsublines@ 233,
 - 326, 371, 404, 411, 442, 451,
 - 463, 470, 482, 516, 563, 565,
 - 1119, 1138, 1211, 1298, 1555, 1559
 - \ifvbox 983, 1075, 1841, 1844, 2445, 2482
 - \ifvoid 2252, 2253
 - \ifwidthliketwocolumns 12
 - \ifwrittenlinesL 2441, 2453
 - \ifwrittenlinesR 2442, 2490
 - \init@series@eledpar 713
 - \initnumbering@sectcmd 745, 759
 - \initnumbering@sectcountR
 - 59, 92, 112, 2183
 - \InputIfFileExists 62
 - \insert@count 539, 1627, 1654
 - \insert@countR 540, 1623, 1643
 - \insert@noterule@ledgroup
 - 2549, 2557, 2565
 - \inserthangingsymbolfalse 996
 - \inserthangingsymbolL ... 1029, 1658
 - \inserthangingsymbolR 1658
 - \inserthangingsymbolRfalse 1088
 - \inserthangingsymbolRtrue 1086
 - \inserthangingsymboltrue 994
 - \insertlines@listR
 - 73, 243, 259, 545, 1412, 1416
 - \inserts@list 842, 1626, 1653
 - \inserts@listR 891, 1407,
 - 1410, 1420, 1434, 1435, 1622, 1642
 - \instanzaLfalse 1947, 2238
 - \instanzaLtrue 783
 - \instanzaRfalse 1948, 2239
 - \instanzaRtrue 806
 - \itemcount@ 1573, 1575,
 - 1580, 1583, 1601, 1603, 1608, 1610
- L**
- \l@d@nums 653, 656, 662, 665
 - \l@d@set 419, 636, 637
 - \l@dbbl@set@language 1781, 1804
 - \l@dbfnote 1617
 - \l@dc@maxchunks 848, 850,
 - 897, 899, 1739, 1749, 1757, 1764
 - \l@dcalc@maxoftwo
 - 2116, 2327, 2467, 2502
 - \l@dcalc@minoftwo .. 2124, 2226, 2327
 - \l@dcalcnun 1232
 - \l@dchecklang 1781
 - \l@dchset@num 293, 296, 419
 - \l@dcsnotetext 1582, 1585
 - \l@dcsnotetext@l 1585, 1602
 - \l@dcsnotetext@r 1585, 1609
 - \l@demptyd@ta 987, 1079
 - \l@dend@stuff 56, 110, 126, 134
 - \l@dgetline@margin 177
 - \l@dgetsidenote@margin 1564
 - \l@dhiddenumberfalse 1001, 1093
 - \l@dhiddenumbertrue 1191
 - \l@dld@ta 1020,
 - 1320, 1332, 1346, 1358, 1367, 1379
 - \l@dleftbox 967,
 - 998, 1013, 1910, 2151, 2152, 2160
 - \l@dlinenumR 225
 - \l@dlsn@te 1022
 - \l@dmake@labelsR 1542
 - \l@dminpagelines
 - 2082, 2125, 2227, 2349, 2370
 - \l@dnumpstartsL 722, 847,
 - 848, 850, 852, 866, 867, 937,

- 938, [1743](#), 1775, 1829, 1866,
1867, 1943, 2092, 2093, 2236, 2457
`\l@dnumpstartsR` 40, 727, 896,
897, 899, 901, 914, 915, 959,
960, [1743](#), 1776, 1832, 1866,
1867, 1944, 2092, 2093, 2237, 2494
`\l@doldbbl@set@language` 1803
`\l@doldselectlanguage` 1802, 1806, 1811
`\l@dpagetrue` 2340, 2389, 2394, 2401, 2406
`\l@dpagetrue` 2340, 2388, 2393, 2400, 2405
`\l@dpagingfalse` 744, 764
`\l@dpagingtrue` 758
`\l@dpairingfalse` 748, 763
`\l@dpairingtrue` 743, 757
`\l@dprintingcolumnfalse` 1945
`\l@dprintingcolumntrue` 1864
`\l@dprintingpagesfalse` 2240
`\l@dprintingpagetrue` 2087
`\l@dpscl` 981, 983,
988, 1006, 1037, 1047, 1050,
1745, 1777, 1829, 1841, 1875,
1881, 1889, 1892, 1894, 1941,
2104, 2112, 2117, 2120, 2128,
2143, 2144, 2146, 2165, 2166,
2234, 2445, 2447, 2450, 2457,
2465–2467, 2469, 2471, 2544, 2599
`\l@dpsclR` 1072, 1075, 1080,
1098, 1746, 1778, 1832, 1844,
1876, 1882, 1900, 1903, 1905,
1942, 2105, 2113, 2121, 2129,
2185, 2186, 2188, 2207, 2208,
2235, 2482, 2484, 2487, 2494,
2500–2502, 2504, 2506, 2552, 2601
`\l@drd@ta` 1030,
1322, 1330, 1348, 1356, 1369, 1377
`\l@dtrightbox` 967,
1090, 1106, 1915, 2193, 2194, 2202
`\l@drsn@te` 1031
`\l@dsamepagetrue` 2340, 2388, 2393, 2400, 2405
`\l@dsamepagetrue` 2340, 2389, 2394, 2401, 2406
`\l@dsetupmaxlinecounts` 1756, 1773
`\l@dsetuprawboxes` [1748](#), 1772
`\l@dskipnumberfalse` 1296
`\l@dskipnumbertrue` 1187
`\l@dskipversenumberfalse` 2163
`\l@dskipversenumberR` [1162](#)
`\l@dskipversenumberRfalse` 2205
`\l@dskipversenumberRtrue` 1189
`\l@dunhbox@line` 1030, 1053, 1056
`\l@dusedbabelfalse` [1780](#), 1799
`\l@dusedbabeltrue` [1780](#), 1801
`\l@duselanguage`
. [1791](#), 1887, 1898, 2140, 2181
`\l@dzeromaxlinecounts` 1756, 1774
`\l@dzeropenalties` 925, 947, 1870, 2099
`\l@luatexbodydir@L` 831, 1044
`\l@luatexbodydir@R` 880
`\l@luatexpardir@L` 830, 1043
`\l@luatexpardir@R` 879
`\l@luatexttdir@L`
. 829, 1026, 1042, 1045
`\l@luatexttdir@R` 878
`\l@prev@nopbR` 47, 2531, 2538, 2539
`\l@prev@pbR` 46, 2530, 2536, 2537
`\l@pscl` [1745](#)
`\l@psclR` [1745](#)
`\labelref@list` 1560
`\labelref@listR` 1540, 1556
`\language` 1782, 1783, 1785–1788
`\last@page@num` 358, 364
`\last@page@numR` [344](#)
`\lastbox` 991, 1083
`\lastskip` 607, 613
`\Lcolwidth` 4, 5, [14](#),
760, 861, 999, 1013, 1023, 1895,
2009, 2030, 2047, 2058, 2074, 2095
`\led@err@BadLeftRightPstarts`
. [23](#), 1867, 2093
`\led@err@LeftOnRightPage` [26](#), 2280
`\led@err@LineationInNumbered` 140
`\led@err@ManyLeftnotes` 1603
`\led@err@ManyRightnotes` 1610
`\led@err@ManySidenotes` 1583
`\led@err@NumberingNotStarted` 84
`\led@err@NumberingShouldHaveStarted`
. 114
`\led@err@NumberingStarted` 37
`\led@err@PendNoPstart` 923, 945
`\led@err@PendNotNumbered` 920, 942
`\led@err@PstartInPstart` 838, 887
`\led@err@PstartNotNumbered` 834, 883
`\led@err@RightOnLeftPage` [26](#), 2289
`\led@err@TooManyPstarts` [20](#), 849, 898
`\led@mess@NotesChanged` 81
`\led@mess@SectionContinued`
. 108, 124, 132

- \led@nopbnumR 2535, [2536](#)
- \led@nopbR 2534, [2536](#)
- \led@pb@setting
 1972, 1980, 2387, 2391,
 2399, 2403, 2414, 2415, 2425, 2426
- \led@pbnumR 2533, [2536](#)
- \led@pbR 2532, [2536](#)
- \led@warn@BadAction 1193
- \led@warn@BadAdvancelineLine 389, 395
- \led@warn@BadAdvancelineSubline .
 375, 381
- \led@warn@BadLineation 166
- \led@warn@BadSetline 626
- \led@warn@BadSetlinenum 634
- \led@warn@DuplicateLabel 1544
- \ledgroupnotesL@false 2560
- \ledgroupnotesL@true 2548
- \ledgroupnotesR@false 2563
- \ledgroupnotesR@true 2556
- \ledllfill 1023
- \lednopb 802
- \lednopbnum 2414, [2532](#)
- \lednopbnumR 2425, [2532](#)
- \lednopbR 802, 2534
- \ledpb 801
- \ledpbnum 2415
- \ledpbnumR 2426, [2532](#)
- \ledpbR 801, [2532](#)
- \ledRcol@false 1109
- \ledRcol@true 1073
- \ledRcolfalse 775, 808
- \ledRcoltrue 792
- \ledrlfill 1030, 2018, 2040
- \ledsavedprintlines [10](#), [1440](#)
- \ledsectnomark 1036
- \ledsectnotoc 1035, 2142, 2184
- \ledstrutL 2152, 2160, [2264](#)
- \ledstrutR 2194, 2202, [2264](#)
- \ledthegoal 2348, 2369, [2431](#)
- \leftlinenumR [225](#), 1320, 1332
- \leftpstartnumL [1341](#)
- \leftpstartnumR [1341](#)
- Leftside (environment) [7](#), [774](#)
- \Leftsidehook 781, [786](#)
- \Leftsidehookend 785, [786](#)
- \letcs 687, 693, 696, 981, 1072
- \line@list 660, 664
- \line@list@stuff 125
- \line@list@stuffR .. 55, 109, 133, [589](#)
- \line@listR . 76, [243](#), 258, 565, 651, 655
- \line@margin 182, 1351
- \line@marginR [175](#), 1325, 1372
- \line@num . 361, 393, 394, 396, 414,
 425, 426, 454, 572, 1144, 1558, 2460
- \line@numR 48,
 232, 239, 298, 332, 351, 387,
 388, 390, 407, 421, 422, 445,
 558, 562, 576, 1125, 1205, 1214,
 1303, 1305, 1307, 1308, 1554, 2497
- \lineation 171, 172, 803
- \lineation* [7](#), [171](#)
- \lineationR [7](#), [138](#), 173, 803
- \linenum@out ... 604, 610, 616, 622,
 629, 637, 642, 646, 1852, 1930, 2321
- \linenum@outR
 . [586](#), 592, 594, 599, 600, 606,
 609, 615, 621, 628, 636, 641,
 645, 1857, 1934, 2324, 2532–2535
- \linenumberlist 1304, 1308
- \linenumincrement [7](#), [196](#)
- \linenumincrement* [7](#), [196](#)
- \linenummargin [175](#)
- \linenumr@p 1443, 1447, 1554, 1558
- \linenumrepR [222](#), 232
- \linenumsep
 . 227, 229, 1388, 1391, 1400, 1403
- \linesinpar@listL
 [250](#), 268, 573, 2295, 2298
- \linesinpar@listR
 [250](#), 262, 577, 2301, 2304
- \lineskip 1492, 1535
- \linesonpage@listL 269, 581, 2308, 2311
- \linesonpage@listR 263, 584, 2314, 2317
- \list@clear
 . 258–265, 268, 269, 271, 842, 891
- \list@clearing@reg 267
- \list@create 243–248,
 250–252, 682, 718, 719, 1407, 1540
- \list@pstartL@pc .. [718](#), 721, 731, 732
- \list@pstartR@pc .. [718](#), 726, 737, 738
- \listbreak 2255, 2259
- \listxadd 366, 2536–2539
- \lock@disp 1264, 1268, 1273
- \lock@off 503, 504, [512](#), 645, 646
- \lock@on 641, 642
- \luatexbodydir 831, 880, 1044
- \luatexpardir 830, 879, 1043
- \luatextextdir .. 829, 878, 1026, 1042

M

\managestanza@modulo 1715
 \maxchunks 3, 1739
 \maxlinesinpar@list 250, 271
 \memorydump 9, 780, 797
 \memorydumpL 119, 780
 \memorydumpR 119, 797
 \message 54
 \multiply 1238, 2012, 2061

N

\n@num 540
 \namebox 983, 988, 1075,
 1080, 1723, 1841, 1844, 2445, 2482
 \NeedsTeXFormat 2
 \new@line 1053, 1056
 \new@lineL 603, 1028
 \new@lineR 605
 \newbool 673, 676
 \newbox 813, 967, 968, 1724, 2247, 2250
 \newcommandx .. 821, 870, 918, 940, 1703
 \newcounter 92–
 95, 187, 189, 191, 193, 816, 818
 \newhookcommand@series 706, 709
 \newif 6, 33, 136, 137,
 587, 772, 773, 965, 966, 1163,
 1395, 1658, 1780, 1826, 1838,
 1970, 1971, 2340, 2342, 2441, 2442
 \newlength 1998, 2001
 \newmarks 2540–2542
 \newnamebox 1723, 1751, 1752
 \newnamecount 1734, 1759
 \newsavebox 1861, 1862
 \newseries@eledpar 670, 714
 \newwrite 586, 2529
 \next@absline
 1973, 1975, 1977, 2392–2394
 \next@abslineR
 1974, 1976, 1978, 2404–2406
 \next@action 288
 \next@actionline 285, 287
 \next@actionlineR
 . 277, 279, 1156, 1201, 1223, 1225
 \next@actionR 280, 1157,
 1202, 1203, 1208, 1209, 1217, 1226
 \next@insert 843
 \next@insertR
 892, 1411, 1414, 1416, 1419, 1423
 \next@page@num 365, 437
 \next@page@numR 52, 301, 303, 355, 434

\noindent 867, 915, 938, 960
 \normal@page@break 366
 \normal@page@breakR 45
 \normal@pars 69, 846, 895
 \normalbfnoteX 1631
 \notefontsetup 2586
 \noteschanged@true
 74, 77, 652, 661, 1413
 \notesXwidthliketwocolumns 4
 \num@lines 926, 1871, 2100
 \num@linesR 812, 948, 1872, 2101
 \numberedpar@true 862, 910
 \numberingRfalse 68
 \numberingRtrue 42, 103, 129
 \numberingtrue 121
 \numberpstartfalse 10
 \numberpstarttrue 10
 \numdef 688, 1037, 1601, 1608,
 1973, 1974, 2392, 2404, 2599, 2601
 \numgdef 1573, 1580, 2413, 2424
 \numlabfont 232
 \numpagelinesL 2082,
 2149, 2173, 2177, 2349, 2415, 2619
 \numpagelinesR
 2082, 2191, 2215, 2219, 2370, 2426

O

\old@footnote 696, 700
 \old@otherlanguage 1816
 \old@startstanza .. 782, 783, 805, 806
 \oldchapter 755, 765
 \one@line .. 988, 991, 1030, 1053, 1056
 \one@lineR 812, 1080, 1083
 \onlysideX 6
 \onlyXside 6
 \openout 63, 594, 600
 \otherlanguage 1816, 1817

P

\p@pstartL 864
 \p@pstartR 912
 \PackageError 19
 \page@action 302, 431, 551
 \page@num 283, 363, 1353
 \page@numR 254, 275, 353,
 557, 562, 1203, 1327, 1374, 1596
 \pagebreak 1987
 \pagegoal 2439
 \Pages 5, 2086
 pages (environment) 5, 742

- `\pagetotal` 2154, 2156, 2196,
2198, 2271, 2277, 2286, 2348, 2369
 - `pairs` (environment) 3, 742
 - `\paperwidth` 1052, 1055
 - `\par@line` 927, 1873, 2102
 - `\par@lineR` 812, 949, 1874, 2103
 - `\parbox` 1895, 1906
 - `\parledgroup@` 990, 1082, 2540, 2566
 - `\parledgroup@beforenotes@save` . .
. 935, 957, 2626
 - `\parledgroup@beforenotesL` 2624
 - `\parledgroup@beforenotesR` 2624
 - `\parledgroup@correction@notespacing`
. 2159, 2201, 2613
 - `\parledgroup@correction@notespacing@final`
. 2475, 2510, 2594
 - `\parledgroup@correction@notespacing@init`
. 2178, 2220, 2589, 2597
 - `\parledgroup@notes@endL`
. 2448, 2451, 2474, 2559
 - `\parledgroup@notes@endR`
. 2485, 2488, 2509, 2562
 - `\parledgroup@notes@startL`
. 990, 2543, 2559
 - `\parledgroup@notes@startR`
. 1082, 2543, 2559
 - `\parledgroup@notespacing@correction`
. 2584, 2616–2618
 - `\parledgroup@notespacing@correction@accumulated`
. 2590, 2596, 2617
 - `\parledgroup@notespacing@correction@modulo`
. 2591, 2618–2620
 - `\parledgroup@notespacing@set@correction`
. 2089, 2584
 - `\parledgroup@series`
. 2541, 2545, 2546,
2553, 2554, 2569, 2570, 2576, 2577
 - `\parledgroup@type` 2542,
2545, 2546, 2553, 2554, 2567, 2574
 - `\parledgroupnotespacing` 2583, 2586
 - `\parledgroupseries@` 2540
 - `\parledgrouptrue` 11
 - `\parledgrouptype@` 2540
 - `\pausenumbering` 795
 - `\pausenumberingR` 102, 795
 - `\pend` 7, 779, 800, 839, 1705
 - `\pendL` 779, 918
 - `\pendR` 800, 888, 940
 - `\pretocmd` 2517
 - `\prev@abslineverse`
. 2413–2415, 2424–2426
 - `\prev@nopbR` 2530
 - `\prev@pbR` 2530
 - `\prevgraf`
. 926, 948, 1871, 1872, 2100, 2101
 - `\print@columnseparator` 1914, 1950
 - `\print@eledsectionL` 1033, 1895, 2147
 - `\print@eledsectionR` 1113, 1906, 2189
 - `\print@last@after@pendLfalse` 2361
 - `\print@last@after@pendLtrue` 2477
 - `\print@last@after@pendRfalse` 2382
 - `\print@last@after@pendRtrue` 2512
 - `\print@lineL` 1008, 1018
 - `\print@lineR` 1100, 1113
 - `\print@notesX` 2109
 - `\print@notesX@forpages` 1453, 2109
 - `\print@Xnotes` 2108
 - `\print@Xnotes@forpages` 1453, 2108
 - `\printlines` 1451
 - `\printlinesR` 10, 1440
 - `\ProcessOptions` 13
 - `\protected@csxdef` 698
 - `\protected@edef` 863, 911
 - `\protected@write` 1785
 - `\protected@xdef` 1640, 1651
 - `\ProvidesPackage` 3
 - `\pst@rtedLfalse` 32
 - `\pst@rtedLtrue` 122, 844
 - `\pst@rtedRfalse` 41, 71
 - `\pst@rtedRtrue` 106, 130, 893
 - `\pststart` 7, 21, 25, 777, 799, 1699
 - `\pststartinfootnote` 147, 155, 163
 - `\pststartL` 777, 815
 - `\pststartnumfalse` 1388, 1393
 - `\pststartnumRfalse` 1400, 1405
 - `\pststartnumRtrue` 1396, 1880, 2507
 - `\pststartnumtrue` 1879, 2472
 - `\pststartR` 799, 815
- R**
- `\Rcolwidth` 4, 5,
14, 761, 909, 1091, 1106, 1906,
2010, 2031, 2048, 2059, 2075, 2096
 - `\read@linelist` 256, 590
 - `\rem@inder` 1308, 1310–1312
 - `\RequirePackage` 5
 - `\resetprevline@` 1931, 1935, 2461, 2498
 - `\restore@pststartL@pc`
. 730, 1883, 2130, 2473

- \restore@pstartR@pc 730, 1884, 2131, 2508
 - \resumenumbering 796
 - \resumenumberingR 102, 796
 - \rightlinenumR 225, 1322, 1330
 - \rightpstartnumL 1341
 - \rightpstartnumR 1341
 - Rightside (environment) 7, 791
 - \Rightsidehook 786, 804
 - \Rightsidehookend 786, 809
 - \rlap 1322, 1330, 1348, 1356, 1369, 1377
 - \Rlineflag 9, 220, 232, 1443, 1447, 1546
 - \rule 1990
- S**
- \savebox 1895, 1906
 - \sc@n@list 1309, 1311
 - \secdef 770
 - \section@num 123–125
 - \section@numR 30, 43, 54, 55, 62, 63, 107–109, 131–133
 - \select@language 1783, 1785–1788, 1822
 - \selectlanguage 1791
 - \set@line 649
 - \set@line@action 295, 400, 409, 416, 439, 553
 - \setl@dlp@rbox 1589, 1604, 1606
 - \setl@drp@rbox 1591, 1599, 1611
 - \setline 624
 - \setlinenum 632
 - \setnamebox 852, 901, 1723
 - \setnotepositionliketwocolumns@C 2004
 - \setnotepositionliketwocolumns@L 2004
 - \setnotepositionliketwocolumns@R 2004
 - \setnotespositionliketwocolumns@C 2043
 - \setnotespositionliketwocolumns@L 2021
 - \setnotespositionliketwocolumns@R 2070
 - \setpositionliketwocolumns@C ... 2004, 2038
 - \setpositionliketwocolumns@L ... 2004, 2017
 - \setpositionliketwocolumns@R ... 2004, 2066
 - \setprintlines 1441
 - \setwidthliketwocolumns@C 2004, 2025
 - \setwidthliketwocolumns@L 2004, 2004
 - \setwidthliketwocolumns@R 2004, 2055
 - \shiftedpstartsfalse 8
 - \shiftedpstartstrue 7, 9, 10
 - \shiftedversesfalse 8
 - \shiftedversestrue 7
 - \sidenote@margin 1566, 1569
 - \sidenote@marginR 1563, 1594
 - \sidenotecontent@ 1572, 1577, 1578, 1589, 1591, 1599, 1600, 1604, 1606, 1607, 1611
 - \sidenotemargin 1563
 - \sidenotemargin* 1563
 - \sidenotesep 1578
 - \skip 1469, 1477, 1513, 1521, 2569, 2576
 - \skip@lockoff 504, 512
 - \skipnumbering 10, 649
 - \smash 1990
 - \splitbotmarks 2566, 2567, 2569, 2570, 2574, 2576, 2577
 - \splitfirstmarks 990, 1082, 2545, 2546, 2553, 2554
 - \splittopskip .. 985, 1077, 1489, 1532
 - \stanza@count 1683, 1696, 1710
 - \stanza@hang 1685, 1718
 - \stanza@modulo 1683, 1713
 - \stanzaindentbase 1663, 1672, 1711, 1714
 - \startlock 640
 - \startstanzahook 1681
 - \startsub 607
 - \sub@action 311, 460, 552
 - \sub@change 53, 305, 306, 312
 - \sub@lock 1139
 - \sub@lockR 50, 320, 322, 324, 327, 478, 484, 485, 487, 488, 518, 519, 521, 1120, 1179, 1181, 1182, 1184, 1245, 1285, 1287, 1289
 - \sub@off 615, 616
 - \sub@on 609, 610
 - \subline@num 234, 361, 379, 380, 382, 412, 452, 1140, 1145, 1559
 - \subline@numR 235, 239, 328, 332, 351, 373, 374, 376, 405, 443, 559, 563, 1121, 1126, 1205, 1212, 1299, 1300, 1555
 - \sublinenumincrement 7, 196
 - \sublinenumincrement* 7, 196
 - \sublinenumr@p . 1444, 1448, 1555, 1559

<code>\sublinenumrepR</code>	222, 235	<code>\unvnamebox</code>	1723
<code>\sublines@false</code>	51, 309, 1169	<code>\usebox</code>	1912, 1917
<code>\sublines@true</code>	307, 1167	<code>\usenamecount</code> 1684, 1690, <u>1734</u> , 1766,	
<code>\sublock@disp</code>	1247, 1251, 1256	2117, 2447, 2450, 2467, 2469,	
<code>\sw@list@inedtextR</code>	248, 265	2484, 2487, 2502, 2504, 2544, 2552	
<code>\sw@listR</code>	247, 264		
<code>\symplinenum</code>	1443		
<code>\sza@penalty</code>	1691, 1695		
		V	
T		<code>\value</code>	1709
<code>\temp</code>	2007, 2008, 2011–2014,	<code>\vbadness</code>	984, 1076
2027, 2028, 2033–2035, 2044,		<code>\vbfnoteX</code>	1641, 1652
2046, 2049–2052, 2056, 2057,		<code>\vbox</code>	852,
2060–2063, 2071, 2073, 2076–2079		901, 1490, 1533, 1895, 1906, 2519	
<code>\temp@</code>	1037, 1038	<code>\vl@dbfnote</code>	1621, 1625
<code>\temp@spacing</code>	2586, 2587	<code>\vsplit</code>	988, 1080, 1491, 1534
<code>\tempa</code> 2045, 2047–2049, 2072, 2074–2076			
<code>\textwidth</code> 15, 17, 760, 761, 2095, 2096		W	
<code>\theledlanguageL</code> . . . <u>1791</u> , 1887, 2140		<code>\widthliketwocolumns</code>	4
<code>\theledlanguageR</code> . . . <u>1791</u> , 1898, 2181		<code>\widthliketwocolumnstrue</code>	12
<code>\thepage</code>	604, 606	<code>\WithSuffix</code>	171, 216–219, 1563
<code>\thepstart</code>	778, 798	<code>\writtenlinesLfalse</code>	2106, 2458
<code>\thepstartL</code>		<code>\writtenlinesLtrue</code>	2455
10, 778, 817, 857, 864, 1387, 1392		<code>\writtenlinesRfalse</code>	2107, 2495
<code>\thepstartR</code>		<code>\writtenlinesRtrue</code>	2492
10, 798, 819, 905, 912, 1399, 1404			
<code>\thisfootnote</code> . . 1640, 1641, 1651, 1652		X	
<code>\thr@@</code> . . 487, 496, 519, 526, 1174, 1182		<code>\xifinlist</code>	1006,
<code>\togglefalse</code>	686	1038, 1098, 1889, 1900, 2146, 2188	
<code>\toggletrue</code>	684	<code>\xifinlistcs</code>	1975–
<code>\topskip</code>	2271	1978, 1981–1984, 2388, 2389,	
		2393, 2394, 2400, 2401, 2405, 2406	
U		<code>\Xnoteswidthliketwocolumns</code>	4
<code>\unexpanded</code>	963	<code>\xpg@main@language</code>	1823, 1824
<code>\unhbox</code>	1728,	<code>\xright@appenditem</code>	
1910, 1915, 2152, 2160, 2194, 2202		433, 434, 436, 437, 441,
<code>\unhnamebox</code>	<u>1723</u>	448, 450, 457, 462, 464, 466,	
<code>\unless</code>	146,	469, 471, 473, 481, 483, 492,	
154, 162, 671, 675, 679, 705, 708		515, 517, 524, 545, 561, 573,	
<code>\unvbox</code> . . . 991, 1083, 1493, 1536, 1730		577, 581, 584, 689, 721, 726,	
		1554, 1558, 1621, 1625, 1641, 1652	
		<code>\xspace</code>	701

Change History

v0.1.		lated code	67
General: First public release	1	Fix babel problems	1
v0.2.		<code>\Columns:</code> Added <code>\l@dcchecklang</code>	
General: Added section of babel re-		and <code>\l@duselanguage</code> to	

\Columns	71	v0.3.a.	
\Pages: Added \l@duselanguage		General: Minor \linenummargin	
to \Pages	78	fix	1
v0.3.		\line@marginR: Don't just	
General: Added \do@lineLhook		set \line@marginR in	
and \do@lineRhook	47	\linenummargin	20
Reorganize for ledarab	1	v0.3.b.	
\affixline@numR: Changed		General: Improved parallel page	
\affixline@numR to match new		balancing	1
eledmac	51	\Pages: Added \l@dminpagelines	
\do@actions@nextR: Used		calculation for succeeding page	
\do@actions@fixedcode in		pairs	80
\do@actionsR	49	v0.3.c.	
\do@lineL: Added \do@lineLhook		General: Compatibilty with Poly-	
to \do@lineL	45	glossia	1
Simplified \do@lineL by using		v0.4.	
macros for some common code	45	General: No more ledparpatch. All	
\do@lineR: Changed \do@lineR		patches are now in the main	
similarly to \do@lineL	47	file.	1
Leftside: Added hooks into Left-		v0.5.	
side environment	39	General: Corrections about	
\flag@end: Removed extraneous		\section and other titles in	
spaces from \flag@end	32	numbered sections	1
\ifledRcol: Moved \ifl@dpairing		v0.6.	
to eledmac	15	General: Be able to us \chapter in	
\ifpst@rtedR: Moved \ifpst@rtedL		parallel pages.	1
to eledmac	16	v0.7.	
\l@dlinenumR: Simplified		General: Option 'shiftedverses'	
\leftlinenumR and \rightlinenumR		which make there is no blank	
by introducing \l@dlinenumR	22	between two parallel verses with	
\l@dnumpstartsR: Moved		inequal length.	1
\l@dnumpstartsL to eledmac	66	v0.8.	
\ledsavedprintlines: Simplified		General: Possibility to have a sym-	
\printlinesR by using		bol on each hanging of verses,	
\setprintlines	56	like in the french typogra-	
\ledstrutR: Added \ledtrutL and		phy. Redefine the commande	
\ledstrutR	81	\hangingsymbol to define the	
\normalbfnoteX: Removed		character.	1
extraneous spaces from		v0.9.	
\normalbfnoteX	63	General: Possibility to number	
\Pages: Added \ledstrutL to		\pstart.	10
\Pages	78	Possibility to number the	
Added \ledstrutR to \Pages	79	pstart with the commands	
\Rightsidehookend: Added		\numberpstarttrue.	1
\Leftsidehook, \Leftsidehookend,		\ifledRcol: Moved \iflledRcol	
\Rightsidehook and \Rightsidehookend	39	and \ifnumberingR to eledmac	15
v0.9.1.		v0.9.1.	
\sublinenumrepR: Added		General: The numbering of the	
\linenumrepR and \sublinenumrepR		pstarts restarts on each	
	22	\beginnumbering.	1

v0.9.2.	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	v1.0.	General: Compatibility with <code>eledmac</code> . Change name to <code>eledpar</code> . . .	1
				Debug in lineation by <code>pstart</code> . .	19
v0.9.3.	General: <code>\thepstartL</code> and <code>\thepstartR</code> use now <code>\bfseries</code> and not <code>\bf</code> , which is deprecated and makes conflicts with memoir class.	1	v1.0.1.	General: Correction on <code>\numberonlyfirstinline</code> with lineation by <code>pstart</code> or by page.	1
v0.10.	General: <code>\edlabel</code> commands on the right side are now correctly indicated.	1	v1.1.	General: <code>Shiftedverses</code> becomes <code>shiftedpstarts</code>	1
	<code>\edlabel</code> commands which start a paragraph are now put in the right place.	1		<code>\pstartR</code> : Add <code>\labelpstarttrue</code> (from <code>eledmac</code>).	40
v0.11.	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>).	45	v1.1.1.	<code>\pstartR</code> : Correct <code>\pstartR</code> bug introduced by 1.1.	40
	Lineation can be by <code>pstart</code> (like in <code>eledmac 0.15</code>).	19	v1.1.2.	<code>\affixside@noteR</code> : Remove spurious space between line number and line content	62
	New management of hangingsymbol insertion, preventing undesirable insertions.	64	v1.2.	General: Support for <code>\led<section></code> commands in parallel texts. . .	1
	<code>\affixline@numR</code> : Changed <code>\affixline@numR</code> to allow to disable line numbering (like in <code>eledmac 0.15</code>).	51	v1.2.1.	<code>\initnumbering@sectcountR</code> : For the right section, the counter is defined only once.	18
	<code>\Columns</code> : Line numbering by <code>pstart</code>	72	v1.3.	<code>\edtext</code> : Manage RTL language. . .	34
	<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>).	86	v1.3.1.	<code>\l@dbfnote</code> : Compatibility of standard footnotes with <code>eledmac</code> when theses footnotes contain any commands.	63
	<code>Pstart</code> number can be printed in side	87	v1.3.2.	General: Debug with some classes. .	1
	<code>\inserthangingsymbolR</code> : Prevent the column separator for hanging verse from shifting	64	v1.3.3.	General: Debugging the left notes of the right column.	62
v0.12.	General: New management of hangingsymbol insertion, preventing undesirable insertions.	64		<code>\l@dbfnote</code> : Spurious space with footnote in right column. . . .	63
			v1.3.4.	General: Allow use of commands in sidenotes, as introduced by <code>eledmac 1.0</code>	62
			v1.3.5.	<code>\normalbfnoteX</code> : Allows one to redefine <code>\thefootnoteX</code> with <code>alph</code> when some packages are loaded.	63

v1.4.	General: Added <code>\do@insidelineLhook</code> and <code>\do@insidelineRhook</code> . . .	47	Add, as in <code>eledmac</code> , option to insert something after <code>\pends / verses</code>	1
v1.4.1.	<code>\normalbfnoteX</code> : Fix bug with normal familiar footnotes when mixing RTL and LTR text. . .	63	Add, as in <code>eledmac</code> , option to insert something between <code>\pstarts / verse</code>	1
	<code>astanza</code> : Enable the use of <code>stanzaindent</code> s repetition within <code>astanza</code> environment.	64	Change <code>\do@lineR</code> and <code>\do@lineR</code> to allow new sectioning commands.	45
v1.4.3.	General: Corrects a false hanging verse when a verse is exactly the length of a line.	1	Compatibility with <code>musixtex</code> . . .	1
	<code>\inserthangingsymbolR</code> : Hanging verse is no longer automatically flush right.	64	Debug <code>eledmac</code> sectioning command after using <code>\resumenumbering</code>	1
	<code>\pendL</code> : Spurious spaces in <code>\pendL</code>	43	New sectioning commands, as in <code>eledmac</code>	14
	<code>\pendR</code> : Spurious spaces in <code>\pstartR</code>	43	<code>\Columns</code> : Modify <code>\Columns</code> to enable to add section's title. . .	70
	<code>\pstartR</code> : Spurious spaces in <code>\pstartL</code> and <code>\pstartR</code>	40	Suppress <code>\l@dchecklang</code> from <code>\Columns</code>	71
v1.5.0.	General: Add, as in <code>eledmac</code> , features to manage page breaks. . .	1	<code>\l@dchecklang</code> : Suppress <code>\l@dchecklang</code> which didn't work and was not logical, because both columns could have the same language but not the main language of the document.	68
	<code>\sublinenumincrement*</code> : Add starred version of <code>\firstlinenum</code> , <code>\linenumincrement</code> , <code>\firstsublinenum</code> , <code>\sublinenumincrement</code> to change both Left and Right-side.	21	<code>\Pages</code> : Modify <code>\Pages</code> to enable to add section's title.	76
v1.6.0.	General: Add tool and documentation for parallel ledgroups . . .	12	<code>\pendL</code> : As in <code>eledmac</code> , <code>\pendL</code> can have an optional argument. . .	43
v1.7.0.	General: Add, as in <code>eledmac</code> , features to make crossrefs with <code>pstart</code> numbers.	1	<code>\pendR</code> : As in <code>eledmac</code> , <code>\pendR</code> can have an optional argument. . .	43
v1.8.0.	General: <code>\beginnumbering</code> is defined only on <code>eledmac</code> , not on <code>eledpar</code>	16	<code>\print@columnseparator</code> : Move some code of <code>\Columns</code> to <code>\print@columnseparator</code> . . .	73
	<code>\l@dlsnote</code> , <code>\l@dlsnote</code> and <code>\l@dcsnote</code> defined only one time, in <code>eledmac</code>	62	<code>\pstartR</code> : As in <code>eledmac</code> , <code>\pendL</code> and <code>\pendR</code> can have an optional argument.	40
	Add <code>\beforecolumnseparator</code> and <code>\aftercolumnseparator</code> . . .	4	<code>\sidenotemargin*</code> : <code>\sidenotemargin</code> is now directly defined in <code>eledmac</code> to be able to manage <code>eledpar</code>	61
	Add <code>\columnspan</code>	4	Add <code>\sidenotemargin*</code>	61
	Add, as in <code>eledmac</code> , new system of sectioning commands.	1	<code>\theledlanguageR</code> : Correct left/right language setting with <code>polyglossia</code>	69
			v1.8.1.	
			<code>\do@lineL</code> : Fix a bug with critical notes at the beginning of a page, (maybe added by v1.8.0) (?). .	45

<code>\do@lineR</code> : Fix a bug with critical notes at the beginning of a page, added by v1.8.0 (?).	47	columns (this bug was added in 1.8.2).	1
v1.8.2.		<code>\Pages</code> : Debug wrong pages splitting when no optional argument is used in last <code>\pend</code> (bug was added in v1.8.3).	76
General: Debug <code>\eledxxx</code> with some paper sizes	1	Debug wrong parallel pages synchronization when an <code>\edtext</code> falls across two pages.	76
Debug left and side note (bugs added by 1.8.0)	1	v1.10.1.	
<code>\eledpar@error</code> : Errors specific to <code>eledpar</code> send to <code>eledpar</code> handbook	16	<code>\line@list@stuffR</code> : Revert modification of 1.4.2, which makes bugs with numbering. Leave vertical mode to solve spurious space before <code>minipage</code>	32
<code>\flag@end</code> : <code>\flag@start</code> and <code>\flag@end</code> are now defined only one time for <code>eledmac</code> and <code>eledpar</code>	32	v1.11.0.	
<code>\lineation*</code> : Add <code>\lineation*</code>	20	General: Compatibility of standard footnotes with some <code>biblatex</code> styles.	1
v1.8.3.		<code>\edtext</code> : <code>\critext</code> and <code>\edtext</code> are now defined only in <code>eledmac</code>	34
General: Add <code>\noeledxxx</code> , as in <code>eledmac</code>	1	v1.12.0.	
<code>\doinsidelineRhook</code> : Added <code>\dolineLhook</code> , <code>\dolineRhook</code> , <code>\doinsidelineLhook</code> and <code>\doinsidelineRhook</code>	46	General: Compatibility with <code>Lua^AT_EX</code> RTL languages.	1
<code>\Pages</code> : Debug blank pages when using optional argument in the last <code>\pend</code>	76	<code>\Columns</code> : Add <code>\l@dprintingcolumnstrue</code>	70
<code>\resumenumberingR</code> : Debug <code>\resumenumberingR</code>	18	<code>\edlabel</code> : <code>\edlabel</code> and <code>\edindex</code> works now with <code>hyperref</code> when using <code>eledpar</code>	61
v1.9.0.		<code>\edlabel</code> is now defined only one time for both <code>eledmac</code> and <code>eledpar</code>	61
General: Add <code>\AtBeginPairs</code> macro.	3	<code>\Pages</code> : Add <code>\l@dprintingpagestrue</code>	76
Compatibility with <code>\Xnoteswidthliketwocolumns</code> and <code>\notesXwidthliketwocolumns</code>	1	<code>\print@eledsectionL</code> : Compatibility with <code>Lua^AT_EX</code> RTL languages.	46
<code>\ifwidthliketwocolumns</code> : Added <code>widthliketwocolumns</code> option	15	<code>\print@eledsectionR</code> : Compatibility with <code>Lua^AT_EX</code> RTL languages.	48
<code>\theledlanguageR</code> : Debug left/right language switching with <code>polyglossia</code> . Don't write in <code>.aux</code> file when setting left/right lines.	69	<code>\print@olineL</code> : Compatibility with <code>Lua^AT_EX</code> RTL languages.	46
v1.9.1.		v1.12.1.	
<code>\ifledRcol</code> : Moved <code>\ifl@dpadding</code> to <code>eledmac</code>	15	<code>\print@eledsectionL</code> : Fixes bug with <code>Lua^AT_EX</code> RTL <code>\eledsection</code>	46
v1.10.0.		v1.13.0.	
General: Compatibility with <code>\AtEveryPstart</code> and <code>\AtEveryPend</code>	1	General: Fix bug in <code>shiftedpstarts</code> when size difference between <code>pstarts</code> is very important.	1
Restore critical notes in <code>\eledsection</code> in parallel			

With parallel pages, long notes can now flow from the Left to the right side and from the Right to the left side.	1	<code>\eledsection</code> and similar com- mands for RTL texts with Lua [®] TeX.	1
<code>\clearl@drighthouse</code> : Use <code>\newpage</code> instead of <code>\clearpage</code>	82	The <code>\newifs</code> are not followed by boolean values set to false, be- cause it is the TeX default set- ting.	1
<code>\ifledRcol</code> : Remove false boolean settings which are not needed.	15	v1.15.0. General: Add <code>\AtEveryPstartCall</code>	1
<code>\Pages</code> : Prevent false overfull hboxes when using <code>\Pages</code> out- side of pages environment.	77	Fix vertical spurious space before right <code>\eledchapter</code> (bug added in v1.13.0).	1
When using <code>shiftedpstarts</code> op- tion, a <code>\l@leftbox</code> with a null height will advance the <code>\pagetotal</code> in any case.	76	Prevent vertical space when using <code>\AtEveryPstart</code> or <code>\AtEveryPend</code> with a command which prints nothing	1
<code>astanza</code> : Enable the use of optional argument of <code>&</code> in <code>astanza</code> envi- ronment.	64	<code>\do@actions@nextR</code> : Add action 1008 and 1009	49
v1.13.1. <code>\correct@footinsX@box</code> : Call <code>\correct@footinsX@box</code> and <code>\correct@Xfootins@box</code> di- rectly in <code>\print@notesX@forpages</code> and <code>\print@Xnotes@forpages</code>	57	<code>\inserthangingsymbolR</code> : Prevent more efficiently the column sep- arator from shifting when a verse is hanging	64
Correct <code>\correct@footinsX@box</code> and <code>\correct@Xfootins@box</code>	57	<code>\lineationR</code> : As <code>\lineation</code> , <code>\lineationR</code> automatically set the <code>\pstartinfootnote</code>	19
<code>\Pages</code> : Prevent false empty page after <code>\Pages</code> (bug added in 1.13.0)	76	<code>\n@num</code> : <code>\n@num</code> defined only one time for both <code>eledmac</code> and <code>eled-</code> <code>par</code>	30
v1.14.0. General: Fix bug with line num- ber position when using		<code>\skipnumbering</code> : <code>\skipnumbering</code> defined only one time for both <code>eledmac</code> and <code>eledpar</code>	34