

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

Peter Wilson  
Herries Press<sup>†</sup>  
Maïeul Rouquette<sup>‡</sup>

## Abstract

The **eledmac** package, which is based on the PLAIN T<sub>E</sub>X 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.

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.

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

## Contents

<b>1 Introduction</b>	<b>3</b>
<b>2 The <b>eledpar</b> package</b>	<b>4</b>
2.1 General . . . . .	4
<b>3 Parallel columns</b>	<b>5</b>
<b>4 Facing pages</b>	<b>6</b>
<b>5 Left and right texts</b>	<b>7</b>
<b>6 Numbering text lines and paragraphs</b>	<b>8</b>

---

\*This file (**eledpar.dtx**) has version number v1.8.0, last revised 2014/08/05.

<sup>†</sup>herries dot press at earthlink dot net

<sup>‡</sup>maieul at maieul dot net

<b>7 Verse</b>	<b>10</b>
<b>8 Side notes</b>	<b>12</b>
<b>9 Parallel ledgroups</b>	<b>12</b>
9.1 Parallel ledgroups and <code>setspace</code> package . . . . .	13
<b>10 Sectioning commands</b>	<b>14</b>
<b>11 Implementation overview</b>	<b>14</b>
<b>12 Preliminaries</b>	<b>14</b>
12.1 Messages . . . . .	15
<b>13 Sectioning commands</b>	<b>16</b>
<b>14 Line counting</b>	<b>18</b>
14.1 Choosing the system of lineation . . . . .	18
14.2 Line-number counters and lists . . . . .	21
14.3 Reading the line-list file . . . . .	22
14.4 Commands within the line-list file . . . . .	23
14.5 Writing to the line-list file . . . . .	31
<b>15 Marking text for notes</b>	<b>34</b>
<b>16 Parallel environments</b>	<b>35</b>
<b>17 Paragraph decomposition and reassembly</b>	<b>37</b>
17.1 Boxes, counters, <code>\pstart</code> and <code>\pend</code> . . . . .	38
17.2 Processing one line . . . . .	41
17.3 Line and page number computation . . . . .	44
17.4 Line number printing . . . . .	47
17.5 Pstart number printing in side . . . . .	49
17.6 Add insertions to the vertical list . . . . .	51
17.7 Penalties . . . . .	51
17.8 Printing leftover notes . . . . .	52
<b>18 Footnotes</b>	<b>53</b>
18.1 Normal footnote formatting . . . . .	53
<b>19 Cross referencing</b>	<b>53</b>
<b>20 Side notes</b>	<b>55</b>
<b>21 Familiar footnotes</b>	<b>56</b>
<b>22 Verse</b>	<b>57</b>

<i>List of Figures</i>	3
<b>23 Naming macros</b>	<b>59</b>
<b>24 Counts and boxes for parallel texts</b>	<b>60</b>
<b>25 Fixing babel</b>	<b>61</b>
<b>26 Parallel columns</b>	<b>63</b>
<b>27 Parallel pages</b>	<b>68</b>
<b>28 Sections' titles' commands</b>	<b>78</b>
<b>29 Page break/no page break, depending on the specific line</b>	<b>78</b>
<b>30 Parallel ledgroup</b>	<b>79</b>
<b>31 The End</b>	<b>82</b>
<b>Appendix A Some things to do when changing version</b>	<b>83</b>
Appendix A.1 Migration to eledpar 1.4.3 . . . . .	83
<b>References</b>	<b>83</b>
<b>Index</b>	<b>83</b>
<b>Change History</b>	<b>93</b>

## List of Figures

### 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 LaTeX. 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 TeX 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 31); 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 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}
...
\Columns
\end{pairs}
```

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.  
**\columnseparator** By default it inserts a vertical rule of width `\columnrulewidth`. As this is initially defined to be 0pt the rule is invisible. For a visible rule between the columns you

could try:

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

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

`\columnspan`

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

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

`\beforecolumnseparator`

`\aftercolumnseparator`

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

- 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 come back to the previous behavior, just set them with a negative value.

## 4 Facing pages

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

`\Lcolwidth`      Within the `pages` environment the lengths `\Lcolwidth` and `\Rcolwidth` are  
`\Rcolwidth`      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.

`\goalfraction`      When doing parallel pages `eledpar` has to guess where TeX is going to put  
pagebreaks 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}`

## 5 Left and right texts

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

`Leftside`      The left text is put within the `Leftside` environment and the right text like-  
`Rightside`      wise in the `Rightside` environment. The number of `Leftside` and `Rightside`  
environments must be the same.

Within these environments you can designate the line numbering scheme(s) to be used. The `eledmac` package originally used counters for specifying the numbering scheme; now both `eledmac`<sup>1</sup> 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.

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

`\pstart`      In a serial (non-parallel) mode, each numbered paragraph, or chunk, is con-  
`\pend`      tained between the `\pstart` and `\pend` macros, and the paragraph is output when  
the `\pend` macro occurs. The situation is somewhat different with parallel type-  
setting 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:

---

<sup>1</sup>when used with `ledpatch` v0.2 or greater.

```

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

```

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

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

If you are using the `babel` package with different languages (via, say, `\selectlanguage`) for the left and right texts it is particularly important to select the appropriate language within the `Leftside` and `Rightside` environments. The initial language selected for the right text is the `babel` package's default. Also, it is the *last* `\selectlanguage` in a side that controls the language used in any notes for that side when they get printed. If you are using multilingual notes then it is probably safest to explicitly specify the language(s) for each note rather than relying on the language selection for the side. The right side language is also applied to the right side line numbers.

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

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

## 6 Numbering text lines and paragraphs

<pre> \beginnumbering \endnumbering </pre>	<p>Each section of numbered text must be preceded by <code>\beginnumbering</code> and followed by <code>\endnumbering</code>, like:</p> <pre> \beginnumbering &lt;text&gt; \endnumbering </pre> <p>These have to be separately specified within <code>Leftside</code> and <code>Rightside</code> environments.</p>
--------------------------------------------	--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------



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:

```
\renewcommand*{\Rlineflag}{}
```

`\printlinesR` 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}}
```

```
\numberstarttrue
\numberstartfalse
\thepstartL
\thepstartR
```

It's possible to insert a number at every `\pstart` command. You must use the `\numberstarttrue` command to have it. You can stop the numerotation with `\numberstartfalse`. 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 `\sza000`' it is because you have forgotten to use `\setstanzaindent` 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.

```
\setstanzaindent{1,0,1,0,1,0,1,0,1,0,1}
\begin{pairs}
\begin{Leftside}
\firstlinenum{2}
\linenumincrement{1}
```

```

\beginnumbering
\begin{astanza}
  \stanzanum{1} First in first stanza &
                Second in first stanza &
                Second in first stanza &
                Third in first stanza &
                Fourth in first stanza &

  \interstanza
  \setline{2}\stanzanum{2} First in second stanza &
                Second in second stanza &
                Second in second stanza &
                Third in second stanza &
                Fourth in second stanza \&

\end{astanza}
...

```

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

```

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

    \strut &
    \stanzanum{2}\advanceline{-1} First in second stanza &
                  Second in second stanza &
                  Second in second stanza &
                  Third in second stanza &
                  Fourth in second stanza \&

  \end{astanza}
  ...

```

`\hangingsymbol`      Like in `eledmac`, you could redefine the command `\hangingsymbol` to insert a character in each hanged line. If you use it, you must run  $\text{\LaTeX}$  two time. Example for the french typographie

```
\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 ledgroups to be single-spaced (not half-spaced or double-spaced), just add to your preamble:

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

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

## 10 Sectioning commands

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

`\eledsectmark` By default, the  $\LaTeX$  marks for header are taken 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}[2014/08/05 v1.8.0 eledmac extension for parallel texts]%
4
```

With the option ‘`shiftedpstarts`’ a long `pstart` on the left side (or in the right side) don’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 shifted stop at every end of pages. The `\shiftedverses` is kept for backward compatibility.

```

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

```

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

```

\ifl@dpairing \ifl@dpairing is set TRUE if we are processing parallel texts and \ifl@dpaging
\ifl@dpaging is also set TRUE if we are doing parallel pages. \ifledRcol is set TRUE if we
\ifledRcol are doing the right hand text. \ifl@dpairing is defined in eledmac.
12 \l@dpairingfalse
13 \newif\ifl@dpaging
14 \l@dpagingfalse
15 \ledRcolfalse

\lcolwidth The widths of the left and right parallel columns (or pages).
\Rcolwidth
16 \newdimen\lcolwidth
17 \lcolwidth=0.45\textwidth
18 \newdimen\Rcolwidth
19 \Rcolwidth=0.45\textwidth
20

```

## 12.1 Messages

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

```

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

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

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

```

## 13 Sectioning commands

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

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

```
31 \newcount\section@numR
32 \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`.

```
33 \pst@rtedLfalse
34 \newif\ifpst@rtedR
35 \pst@rtedRfalse
36
```

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

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



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   \immediate\closeout\eled@sectioningR@out
88 }
89
```

`\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<sup>A</sup>T<sub>E</sub>X counter in `\numberingR`.

```

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

```

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

```

\resumenumberingR 100 \newcommand*{\pausenumberingR}{%
101   \endnumberingR\global\numberingRtrue}

```

```

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

```

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

```

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

```

## 14 Line counting

### 14.1 Choosing the system of lineation

Sometimes you want line numbers that start at 1 at the top of each page; sometimes you want line numbers that start at 1 at each `\pstart`; other times you want line numbers that start at 1 at the start of each section and increase regardless of page breaks. `eledpar` lets you choose different schemes for the left and right texts.

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

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

`\bypstart@Rfalse`

`\ifbypage@R`

`\bypage@Rtrue`

`\bypage@Rfalse`

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

`eledpar` will use the line-of-section system unless instructed otherwise.

```
133 \newif\ifbypage@R
134 \newif\ifbypstart@R
135   \bypage@Rfalse
136   \bypstart@Rfalse
```

`\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`.

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

`\linenummargin` `\linenummargin{<word>}` 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.

```
161 \newcount\line@marginR
162 \renewcommand*{\linenummargin}[1]{%
163   \l@getline@margin{#1}%
164   \ifnum\@l@dtmpcntb>\m@ne
165     \ifledRcol
166       \global\line@marginR=\@l@dtmpcntb
```

```

167 \else
168 \global\line@margin=\@l@dttempcntb
169 \fi
170 \fi}}

```

By default put right text numbers at the right.

```

171 \line@marginR=\@ne
172

```

`\c@firstlinenumR` The following counters tell `eledmac` which right text lines should be printed with 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.

```

173 \newcounter{firstlinenumR}
174 \setcounter{firstlinenumR}{5}
175 \newcounter{linenumincrementR}
176 \setcounter{linenumincrementR}{5}

```

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

```

177 \newcounter{firstsublinenumR}
178 \setcounter{firstsublinenumR}{5}
179 \newcounter{sublinenumincrementR}
180 \setcounter{sublinenumincrementR}{5}
181

```

`\firstlinenum` These are the user's macros for changing (sub) line numbers. They are defined in `eledmac v0.7`, but just in case I have started by `\providing` them. The starred versions are specific to `eledpar`.

`\linenumincrement`

`\firstsublinenum`

`\sublinenumincrement`

```

182 \providecommand*\firstlinenum{}
183 \providecommand*\linenumincrement{}
184 \providecommand*\firstsublinenum{}
185 \providecommand*\sublinenumincrement{}
186 \renewcommand*\firstlinenum[1]{%
187 \ifledRcol \setcounter{firstlinenumR}{#1}%
188 \else \setcounter{firstlinenumR}{#1}%
189 \fi}
190 \renewcommand*\linenumincrement[1]{%
191 \ifledRcol \setcounter{linenumincrementR}{#1}%
192 \else \setcounter{linenumincrementR}{#1}%
193 \fi}
194 \renewcommand*\firstsublinenum[1]{%
195 \ifledRcol \setcounter{firstsublinenumR}{#1}%
196 \else \setcounter{firstsublinenumR}{#1}%
197 \fi}
198 \renewcommand*\sublinenumincrement[1]{%
199 \ifledRcol \setcounter{sublinenumincrementR}{#1}%
200 \else \setcounter{sublinenumincrementR}{#1}%

```

```

201 \fi}
202 \WithSuffix\newcommand\firstlinenum*[1]{\setcounter{firstlinenumR}{#1}\setcounter{firstlinenum}{#1}}
203 \WithSuffix\newcommand\linenumincrement*[1]{\setcounter{linenumincrementR}{#1}\setcounter{linenumincr
204 \WithSuffix\newcommand\firstsublinenum*[1]{\setcounter{subfirstlinenumR}{#1}\setcounter{subfirstlinen
205 \WithSuffix\newcommand\sublinenumincrement*[1]{\setcounter{sublinenumincrementR}{#1}\setcounter{subli

```

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

```

206 \newcommand*\Rlineflag{R}
207

```

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

```

208 \newcommand*\linenumrepR[1]{\@arabic{#1}}
209 \newcommand*\sublinenumrepR[1]{\@arabic{#1}}
210

```

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

```

211 \newcommand*\leftlinenumR{%
212 \l@dlinenumR
213 \kern\linenumsep}
214 \newcommand*\rightlinenumR{%
215 \kern\linenumsep
216 \l@dlinenumR}
217 \newcommand*\l@dlinenumR{%
218 \numlabfont\linenumrepR\line@numR\Rlineflag%
219 \ifsublines@
220 \ifnum\subline@num>\z@
221 \unskip\fullstop\sublinenumrepR\subline@numR}%
222 \fi
223 \fi}
224

```

## 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  
`\subline@numR` marginal line numbering and in notes. The count `\subline@numR` stores a sub-line  
`\absline@numR` 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.

```

225 \newcount\line@numR
226 \newcount\subline@numR
227 \newcount\absline@numR
228

```

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

```
229 \list@create{\line@listR}
230 \list@create{\insertlines@listR}
231 \list@create{\actionlines@listR}
232 \list@create{\actions@listR}
233
```

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

```
234 \list@create{\linesinpar@listL}
235 \list@create{\linesinpar@listR}
236 \list@create{\maxlinesinpar@list}
237
```

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

```
238 \newcount\page@numR
239
```

### 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.

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

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

```
241 \ifledRcol
242 \list@clear{\line@listR}%
243 \list@clear{\insertlines@listR}%
244 \list@clear{\actionlines@listR}%
245 \list@clear{\actions@listR}%
246 \list@clear{\linesinpar@listR}%
247 \list@clear{\linesonpage@listR}
248 \else
249 \list@clearing@reg
250 \list@clear{\linesinpar@listL}%
251 \list@clear{\linesonpage@listL}%
252 \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).

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

Now get the file and interpret it.

```
254 \get@linelistfile{#1}%
255 \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.

```

256 \ifledRcol
257   \global\page@numR=\m@ne
258   \ifx\actionlines@listR\empty
259     \gdef\next@actionlineR{1000000}%
260   \else
261     \glp\actionlines@listR\to\next@actionlineR
262     \glp\actions@listR\to\next@actionR
263   \fi
264 \else
265   \global\page@num=\m@ne
266   \ifx\actionlines@list\empty
267     \gdef\next@actionline{1000000}%
268   \else
269     \glp\actionlines@list\to\next@actionline
270     \glp\actions@list\to\next@action
271   \fi
272 \fi}
273

```

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

## 14.4 Commands within the line-list file

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

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

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

```

274 \newcommand{\@nl@regR}{%
275   \ifx\l@dchset@num\relax \else
276     \advance\absline@numR \@ne
277     \set@line@action
278     \let\l@dchset@num\relax
279     \advance\absline@numR \m@ne
280     \advance\line@numR \m@ne%    % do we need this?
281   \fi
282   \advance\absline@numR \@ne
283   \ifx\next@page@numR\relax \else
284     \page@action

```

```

285 \let\next@page@numR\relax
286 \fi
287 \ifx\sub@change\relax \else
288 \ifnum\sub@change>\z@
289 \sublines@true
290 \else
291 \sublines@false
292 \fi
293 \sub@action
294 \let\sub@change\relax
295 \fi
296 \ifcase\@lockR
297 \or
298 \@lockR \tw@
299 \or\or
300 \@lockR \z@
301 \fi
302 \ifcase\sub@lockR
303 \or
304 \sub@lockR \tw@
305 \or\or
306 \sub@lockR \z@
307 \fi
308 \ifsublines@
309 \ifnum\sub@lockR<\tw@
310 \advance\subline@numR \@ne
311 \fi
312 \else
313 \ifnum\@lockR<\tw@
314 \advance\line@numR \@ne \subline@numR \z@
315 \fi
316 \fi}
317
318 \renewcommand*{\@nl}[2]{%
319 \fix@page{#1}%
320 \ifledRcol
321 \@nl@regR
322 \else
323 \@nl@reg
324 \fi}
325

```

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

```

\fix@page 326 \newcount\last@page@numR
327 \last@page@numR=-10000
328 \renewcommand*{\fix@page}[1]{%
329 \ifledRcol
330 \ifnum #1=\last@page@numR
331 \else
332 \ifbypage@R

```



```

333     \line@numR \z@ \subline@numR \z@
334     \fi
335     \page@numR=#1\relax
336     \last@page@numR=#1\relax
337     \def\next@page@numR{#1}%
338     \fi
339 \else
340     \ifnum #1=\last@page@num
341     \else
342         \ifbypage@
343             \line@num \z@ \subline@num \z@
344             \fi
345             \page@num=#1\relax
346             \last@page@num=#1\relax
347             \def\next@page@num{#1}%
348             \listcsadd{normal@page@break}{\the\absline@num}
349             \fi
350 \fi}
351

```

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

```

352 \renewcommand*{\@adv}[1]{%
353     \ifsublines@
354         \ifledRcol
355             \advance\subline@numR by #1\relax
356             \ifnum\subline@numR<\z@
357                 \led@warn@BadAdvancelineSubline
358                 \subline@numR \z@
359             \fi
360         \else
361             \advance\subline@num by #1\relax
362             \ifnum\subline@num<\z@
363                 \led@warn@BadAdvancelineSubline
364                 \subline@num \z@
365             \fi
366         \fi
367     \else
368         \ifledRcol
369             \advance\line@numR by #1\relax
370             \ifnum\line@numR<\z@
371                 \led@warn@BadAdvancelineLine
372                 \line@numR \z@
373             \fi
374         \else
375             \advance\line@num by #1\relax
376             \ifnum\line@num<\z@
377                 \led@warn@BadAdvancelineLine
378                 \line@num \z@
379             \fi

```

```

380   \fi
381   \fi
382   \set@line@action}
383

```

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

```

384 \renewcommand*{\@set}[1]{%
385   \ifledRcol
386     \ifsublines@
387       \subline@numR=#1\relax
388     \else
389       \line@numR=#1\relax
390     \fi
391     \set@line@action
392   \else
393     \ifsublines@
394       \subline@num=#1\relax
395     \else
396       \line@num=#1\relax
397     \fi
398     \set@line@action
399   \fi}
400

```

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

```

401 \renewcommand*{\l@d@set}[1]{%
402   \ifledRcol
403     \line@numR=#1\relax
404     \advance\line@numR \@ne
405     \def\l@dchset@num{#1}
406   \else
407     \line@num=#1\relax
408     \advance\line@num \@ne
409     \def\l@dchset@num{#1}
410   \fi}
411 \let\l@dchset@num\relax
412

```

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

```

413 \renewcommand*{\page@action}{%
414   \ifledRcol
415     \xright@appenditem{\the\absline@numR}\to\actionlines@listR
416     \xright@appenditem{\next@page@numR}\to\actions@listR
417   \else
418     \xright@appenditem{\the\absline@num}\to\actionlines@list

```

```

419 \xright@appenditem{\next@page@num}\to\actions@list
420 \fi}

```

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

```

421 \renewcommand*{\set@line@action}{%
422 \ifledRcol
423 \xright@appenditem{\the\absline@numR}\to\actionlines@listR
424 \ifsublines@
425 \@l@dttempcnta=-\subline@numR
426 \else
427 \@l@dttempcnta=-\line@numR
428 \fi
429 \advance\@l@dttempcnta by -5000\relax
430 \xright@appenditem{\the\@l@dttempcnta}\to\actions@listR
431 \else
432 \xright@appenditem{\the\absline@num}\to\actionlines@list
433 \ifsublines@
434 \@l@dttempcnta=-\subline@num
435 \else
436 \@l@dttempcnta=-\line@num
437 \fi
438 \advance\@l@dttempcnta by -5000\relax
439 \xright@appenditem{\the\@l@dttempcnta}\to\actions@list
440 \fi}
441

```

**\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.

```

442 \renewcommand*{\sub@action}{%
443 \ifledRcol
444 \xright@appenditem{\the\absline@numR}\to\actionlines@listR
445 \ifsublines@
446 \xright@appenditem{-1001}\to\actions@listR
447 \else
448 \xright@appenditem{-1002}\to\actions@listR
449 \fi
450 \else
451 \xright@appenditem{\the\absline@num}\to\actionlines@list
452 \ifsublines@
453 \xright@appenditem{-1001}\to\actions@list
454 \else
455 \xright@appenditem{-1002}\to\actions@list
456 \fi
457 \fi}
458

```

**\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.

```

459 \newcount\@lockR
460 \newcount\sub@lockR
461
462 \newcommand*\do@lockonR}{%
463   \xright@appenditem{\the\absline@numR}\to\actionlines@listR
464   \ifsublines@
465     \xright@appenditem{-1005}\to\actions@listR
466     \ifnum\sub@lockR=\z@
467       \sub@lockR \@ne
468     \else
469       \ifnum\sub@lockR=\thr@@
470         \sub@lockR \@ne
471       \fi
472     \fi
473   \else
474     \xright@appenditem{-1003}\to\actions@listR
475     \ifnum\@lockR=\z@
476       \@lockR \@ne
477     \else
478       \ifnum\@lockR=\thr@@
479         \@lockR \@ne
480       \fi
481     \fi
482   \fi}
483
484 \renewcommand*\do@lockon}{%
485   \ifx\next\lock@off
486     \global\let\lock@off=\skip@lockoff
487   \else
488     \ifledRcol
489       \do@lockonR
490     \else
491       \do@lockonL
492     \fi
493   \fi}

\lock@off \lock@off adds an entry to the action-code list to turn line number locking off.
\do@lockoff 494
\do@lockoffR 495
\skip@lockoff 496 \newcommand*\do@lockoffR}{%
497   \xright@appenditem{\the\absline@numR}\to\actionlines@listR
498   \ifsublines@
499     \xright@appenditem{-1006}\to\actions@listR
500     \ifnum\sub@lockR=\tw@
501       \sub@lockR \thr@@
502     \else
503       \sub@lockR \z@
504     \fi
505   \else
506     \xright@appenditem{-1004}\to\actions@listR

```

```

507 \ifnum\@lockR=\tw@
508 \@lockR \thr@@
509 \else
510 \@lockR \z@
511 \fi
512 \fi}
513
514 \renewcommand*{\do@lockoff}{%
515 \ifledRcol
516 \do@lockoffR
517 \else
518 \do@lockoffL
519 \fi}
520 \global\let\lock@off=\do@lockoff
521

```

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

```

522 \providecommand*{\n@num}{}
523 \renewcommand*{\n@num}{%
524 \ifledRcol
525 \xright@appenditem{\the\absline@numR}\to\actionlines@listR
526 \xright@appenditem{-1007}\to\actions@listR
527 \else
528 \n@num@reg
529 \fi}
530

```

`\@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`.

```

531 \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.

```

532 \renewcommand*{\@ref}[2]{%
533 \ifledRcol
534 \global\insert@countR=#1\relax
535 \loop\ifnum\insert@countR>\z@
536 \xright@appenditem{\the\absline@numR}\to\insertlines@listR
537 \global\advance\insert@countR \m@ne
538 \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.

```

539 \begingroup
540   \let\@ref=\dummy@ref
541   \let\page@action=\relax
542   \let\sub@action=\relax
543   \let\set@line@action=\relax
544   \let\@lab=\relax
545   #2
546   \global\endpage@num=\page@numR
547   \global\endline@num=\line@numR
548   \global\endsubline@num=\subline@numR
549 \endgroup

```

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

```

550   \xright@appenditem%
551     {\the\page@numR|\the\line@numR|}%
552     \ifsublines@ \the\subline@numR \else 0\fi|%
553     \the\endpage@num|\the\endline@num|}%
554     \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.

```

555   #2
556   \else

```

And when not in right text

```

557   \@ref@reg{#1}{#2}%
558   \fi}

```

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

```

559 \providecommand*\@pend}[1]{%
560 \renewcommand*\@pend}[1]{%
561   \ifbypstart@\global\line@num=0\fi%
562   \xright@appenditem{#1}\to\linesinpar@listL}
563 \providecommand*\@pendR}[1]{%
564 \renewcommand*\@pendR}[1]{%
565   \ifbypstart@R\global\line@numR=0\fi
566   \xright@appenditem{#1}\to\linesinpar@listR}
567

```

`\@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.

```

568 \providecommand*\@lopL}[1]{}
569 \renewcommand*\@lopL}[1]{%
570   \xright@appenditem{#1}\to\linesonpage@listL}
571 \providecommand*\@lopR}[1]{}
572 \renewcommand*\@lopR}[1]{%
573   \xright@appenditem{#1}\to\linesonpage@listR}
574

```

## 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`.

```

575 \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
576 \newif\iffirst@linenum@out@R
577 \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.

```

578 \newcommand*\@line@list@stuffR}[1]{%
579   \read@linelist{#1}%
580   \iffirst@linenum@out@R
581     \immediate\closeout\linenum@outR
582     \global\first@linenum@out@Rfalse
583     \immediate\openout\linenum@outR=#1
584   \else
585     \if@minipage%
586       \if@ledgroup%
587         \closeout\linenum@outR
588         \openout\linenum@outR=#1
589       \else
590         \immediate\closeout\linenum@outR
591         \immediate\openout\linenum@outR=#1\relax
592       \fi
593     \else
594       \closeout\linenum@outR%
595       \openout\linenum@outR=#1\relax%
596     \fi
597   \fi}
598

```

`\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.

```

599 \newcommand*{\new@lineL}{%
600   \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.

```

601 \newcommand*{\new@lineR}{%
602   \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.

```

603 \renewcommand*{\flag@start}{%
604   \ifledRcol
605     \edef\next{\write\linenum@outR{%
606       \string\@ref[\the\insert@countR][ ]}%
607     \next
608   \else
609     \edef\next{\write\linenum@out{%
610       \string\@ref[\the\insert@count][ ]}%
611     \next
612   \fi}
613 \renewcommand*{\flag@end}{%
614   \ifledRcol
615     \write\linenum@outR{ ]}%
616   \else
617     \write\linenum@out{ ]}%
618   \fi}
```

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

```

619 \renewcommand*{\startsub}{\dimen0\lastskip
620   \ifdim\dimen0>0pt \unskip \fi
621   \ifledRcol \write\linenum@outR{\string\sub@on}%
622   \else      \write\linenum@out{\string\sub@on}%
623   \fi
624   \ifdim\dimen0>0pt \hskip\dimen0 \fi}
625 \def\endsub{\dimen0\lastskip
626   \ifdim\dimen0>0pt \unskip \fi
627   \ifledRcol \write\linenum@outR{\string\sub@off}%
628   \else      \write\linenum@out{\string\sub@off}%
629   \fi
630   \ifdim\dimen0>0pt \hskip\dimen0 \fi}
631
```

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

```

632 \renewcommand*{\advanceline}[1]{%
633   \ifledRcol \write\linenum@outR{\string\@adv[#1]}}
```



```
634 \else      \write\linenum@out{\string\@adv[#1]}%
635 \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.

```
636 \renewcommand*{\setline}[1]{%
637   \ifnum#1<\z@
638     \led@warn@BadSetline
639   \else
640     \ifledRcol \write\linenum@outR{\string\@set[#1]}%
641     \else      \write\linenum@out{\string\@set[#1]}%
642   \fi
643 \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.

```
644 \renewcommand*{\setlinenum}[1]{%
645   \ifnum#1<\z@
646     \led@warn@BadSetlinenum
647   \else
648     \ifledRcol \write\linenum@outR{\string\l@d@set[#1]}
649     \else      \write\linenum@out{\string\l@d@set[#1]} \fi
650   \fi}
651
```

**\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.

```
652 \renewcommand*{\startlock}{%
653   \ifledRcol \write\linenum@outR{\string\lock@on}%
654   \else      \write\linenum@out{\string\lock@on}%
655 \fi}
656 \def\endlock{%
657   \ifledRcol \write\linenum@outR{\string\lock@off}%
658   \else      \write\linenum@out{\string\lock@off}%
659 \fi}
660
```

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

```
661 \renewcommand*{\skipnumbering}{%
662   \ifledRcol \write\linenum@outR{\string\n@num}%
663   \advanceline{-1}%
664   \else
665     \skipnumbering@reg
666   \fi}
667
```

## 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` Now we begin `\critext` itself.

We slightly modify the original to make accomodation for when right text is being processed.

```
668 \long\def\critext#1#2/{\leavevmode
669   \begingroup
670     \renewcommand{\@tag}{\no@expands #1}%
671     \set@line
672     \ifledRcol \global\insert@countR \z@
673     \else      \global\insert@count \z@ \fi
674     \ignorespaces #2\relax
675     \@ifundefined{xpg@main@language}{%if not polyglossia
676       \flag@start}%
677     {%if@RTL\flag@end\else\flag@start\fi% be careful on the direction of writing with p
678     }%
679   \endgroup
680   \showlemma{#1}%
681   \ifx\end@lemmas\empty \else
682     \glp\end@lemmas\to\x@lemma
683     \x@lemma
684     \global\let\x@lemma=\relax
685   \fi
686   \@ifundefined{xpg@main@language}{%if not polyglossia
687     \flag@end}%
688     {%if@RTL\flag@start\else\flag@end\fi% be careful on the direction of writing with p
689   }
690 }
```

`\edtext` And similarly for `\edtext`.

```
691 \renewcommand{\edtext}[2]{\leavevmode
692   \begingroup%
693     \renewcommand{\@tag}{\no@expands #1}%
694     \set@line%
```

```

695 \ifledRcol \global\insert@countR \z@%
696 \else \global\insert@count \z@ \fi%
697 \ignorespaces #2\relax%
698 \@ifundefined{xpg@main@language}{%if not polyglossia
699 \flag@start}%
700 {%\ifRTL\flag@end\else\flag@start\fi% be careful on the direction of writing with polyglossia
701 }%
702 \endgroup%
703 \showlemma{#1}%
704 \ifx@end@lemmas\empty \else%
705 \gl@p@end@lemmas\to\x@lemma%
706 \x@lemma%
707 \global\let\x@lemma=\relax%
708 \fi%
709 \@ifundefined{xpg@main@language}{%if not polyglossia
710 \flag@end}%
711 {%\ifRTL\flag@start\else\flag@end\fi% be careful on the direction of writing with polyglossia
712 }%
713 }
714

```

`\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`.

```

715 \renewcommand*{\set@line}{%
716 \ifledRcol
717 \ifx\line@listR\empty
718 \global\noteschanged@true
719 \xdef\l@d@nums{000|000|000|000|000|000|\edfont@info}%
720 \else
721 \gl@p\line@listR\to\@tempb
722 \xdef\l@d@nums{\@tempb|\edfont@info}%
723 \global\let\@tempb=\undefined
724 \fi
725 \else
726 \ifx\line@list\empty
727 \global\noteschanged@true
728 \xdef\l@d@nums{000|000|000|000|000|000|\edfont@info}%
729 \else
730 \gl@p\line@list\to\@tempb
731 \xdef\l@d@nums{\@tempb|\edfont@info}%
732 \global\let\@tempb=\undefined
733 \fi
734 \fi}
735

```

## 16 Parallel environments

The initial set up for parallel processing is deceptively simple.

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

**chapterinpages**

```

736 \newenvironment{pairs}{%
737   \l@dpairingtrue
738   \l@dpagingfalse
739   \initnumbering@sectcmd
740 }{%
741   \l@dpairingfalse
742 }
```

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.

```

743 \newenvironment{pages}{%
744   \let\oldchapter\chapter
745   \let\chapter\chapterinpages
746   \l@dpairingtrue
747   \l@dpagingtrue
748   \initnumbering@sectcmd
749   \setlength{\Lcolwidth}{\textwidth}%
750   \setlength{\Rcolwidth}{\textwidth}%
751 }{%
752   \l@dpairingfalse
753   \l@dpagingfalse
754   \let\chapter\oldchapter
755 }
756 \newcommand{\chapterinpages}{\thispagestyle{plain}%
757                               \global\@topnum\z@
758                               \@afterindentfalse
759                               \secdef\@chapter\@schapter}
760
```

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

**ifinstanzaR**

```

761 \newif\ifinstanzaL
762 \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.

```

763 \newenvironment{Leftside}{%
764   \ledRcolfalse
765   \setcounter{pstartL}{1}
766   \let\pstart\pstartL
767   \let\thepstart\thepstartL
768   \let\pend\pendL
769   \let\memorydump\memorydumpL
```

```

770 \Leftsidehook
771 \let\old@startstanza\@startstanza
772 \def\@startstanza[##1]{\global\instanzaLtrue\old@startstanza[##1]}
773 }{
774 \Leftsidehookend}

```

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

```

\Rightsidehook 775 \newcommand*{\Leftsidehook}{}
\Rightsidehookend 776 \newcommand*{\Leftsidehookend}{}
777 \newcommand*{\Rightsidehook}{}
778 \newcommand*{\Rightsidehookend}{}
779

```

`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.

```

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

```

## 17 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.

## 17.1 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.

```
801 \newcount\num@linesR
```

```
802 \newbox\one@lineR
```

```
803 \newcount\par@lineR
```

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

`\pstartR`

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

We have to have specific left and right `\pstart` when parallel processing; among other things because of potential changes in the linewidth. The `old` counters are used to have the good reset of the `pstart` counters at the beginning of the `\Pages` command.

```
804
```

```
805 \newcounter{pstartL}
```

```
806 \newcounter{pstartLold}
```

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

```
808 \newcounter{pstartR}
```

```
809 \newcounter{pstartRold}
```

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

```
811
```

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

```
813   \if@nobreak%
```

```
814     \let\@oldnobreak\@nobreaktrue%
```

```
815   \else%
```

```
816     \let\@oldnobreak\@nobreakfalse%
```

```
817   \fi%
```

```
818     \@nobreaktrue%
```

```
819   \ifnumbering \else%
```

```
820     \led@err@PstartNotNumbered%
```

```
821     \beginnumbering%
```

```
822   \fi%
```

```
823   \ifnumberedpar@%
```

```
824     \led@err@PstartInPstart%
```

```
825   \pend%
```

826 \fi%

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

```
827 \ifpst@rtedL\else%
828   \setcounter{pstartLold}{\value{pstartL}}%
829   \list@clear{\inserts@list}%
830   \global\let\next@insert=\empty%
831   \global\pst@rtedLtrue%
832 \fi%
833 \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.

```
834 \global\advance\l@dnumpstartsL \one%
835 \ifnum\l@dnumpstartsL>\l@dc@maxchunks%
836   \led@err@TooManyPstarts%
837 \global\l@dnumpstartsL=\l@dc@maxchunks%
838 \fi%
839 \global\setnamebox{l@dLcolrawbox\the\l@dnumpstartsL}=\vbox\bgroup%
840 \ifautopar\else%
841   \ifnumberpstart%
842     \ifsidepstartnum%
843       \else%
844         \thepstartL%
845       \fi%
846     \fi%
847   \fi%
848 \hsize=\Lcolwidth%
849 \numberedpar@true%
850 \iflabeledpstart\protected@edef\@currentlabel%
851   {\p@pstartL\thepstartL}\fi%
```

Dump the optional arguments

```
852 \ifstrempy{#1}%
853 {}%
854 {\csgdef{before@pstartL@the\l@dnumpstartsL}{\noindent#1}}%
855 }

856 \newcommand*{\pstartR}[1][1]{%
857   \if@nobreak%
858     \let\@oldnobreak\@nobreaktrue%
859   \else%
860     \let\@oldnobreak\@nobreakfalse%
861   \fi%
862   \@nobreaktrue%
863   \ifnumberingR \else%
864     \led@err@PstartNotNumbered%
865     \beginnumberingR%
866   \fi%
867   \ifnumberedpar@%
```

```

868   \led@err@PstartInPstart%
869   \pendR%
870 \fi%
871 \ifpst@rtedR\else%
872   \setcounter{pstartRold}{\value{pstartR}}%
873   \list@clear{\inserts@listR}%
874   \global\let\next@insertR=\empty%
875   \global\pst@rtedRtrue%
876 \fi%
877 \begingroup\normal@pars%
878 \global\advance\l@dnumpstartsR \@ne%
879 \ifnum\l@dnumpstartsR>\l@dc@maxchunks%
880   \led@err@TooManyPstarts%
881   \global\l@dnumpstartsR=\l@dc@maxchunks%
882 \fi%
883 \global\setnamebox{\l@dRcolrawbox\the\l@dnumpstartsR}=\vbox\bgroup%
884   \ifautopar\else%
885     \ifnumberpstart%
886       \ifsidepstartnum\else%
887         \thepstartR%
888       \fi%
889     \fi%
890   \fi%
891   \hsize=\Rcolwidth%
892   \numberedpar@true%
893   \iflabelpstart\protected@edef\@currentlabel%
894     {\p@pstartR\thepstartR}\fi%
895   \ifstrempy{#1}%
896     {}
897     {\csgdef{before@pstartR@the\l@dnumpstartsR}{\noindent#1}}%
898 }

```

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

```

899 \newcommand*{\pendL}[1][1]{%
900   \ifnumbering \else%
901     \led@err@PendNotNumbered%
902   \fi%
903   \ifnumberedpar@ \else%
904     \led@err@PendNoPstart%
905   \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.

```

906   \l@dzeropenalties%
907   \endgraf\global\num@lines=\prevgraf\egroup%
908   \global\par@line=0%

```



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

```

909 \endgroup%
910 \ignorespaces%
911 \@oldnobreak%
912 \ifnumberpstart%
913   \addtocounter{pstartL}{1}%
914 \fi
915 \parledgroup@beforenotes@save{L}%
  Dump content of the optional argument.
916 \ifstrempy{#1}%
917   {}%
918   {\csgdef{after@pendL@the\l@dnumpstartsL}{\noindent#1}}%
919 }
```

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

```

920 \newcommandx*{\pendR}[1][1]{%
921   \ifnumberingR \else%
922     \led@err@PendNotNumbered%
923   \fi%
924   \ifnumberedpar@ \else%
925     \led@err@PendNoPstart%
926   \fi%
927   \l@dzeropenalties%
928   \endgraf\global\l@num@linesR=\prevgraf\egroup%
929   \global\l@par@lineR=0%
930   \endgroup%
931   \ignorespaces%
932   \@oldnobreak%
933   \ifnumberpstart%
934     \addtocounter{pstartR}{1}%
935   \fi%
936   \parledgroup@beforenotes@save{R}%
937   \ifstrempy{#1}%
938     {}%
939     {\csgdef{after@pendR@the\l@dnumpstartsR}{\noindent#1}}%
940 }
941
```

## 17.2 Processing one line

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

`\l@dleftbox` A line of left text will be put in the box `\l@dleftbox`, and analogously for a line of right text.

```

942 \newbox\l@dleftbox
943 \newbox\l@drightbox
944
```

```

\countLline We need to know the number of lines processed.
\countRline 945 \newcount\countLline
              946 \countLline \z@
              947 \newcount\countRline
              948 \countRline \z@
              949

\@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 950 \newcount\@donereallinesL
                  951 \newcount\@donetotallinesL
                  952 \newcount\@donereallinesR
                  953 \newcount\@donetotallinesR
                  954

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

          955 \newcommand*\do@lineL{%
          956 \advance\countLline \@ne
          957 \ifvbox\namebox{\l@dLcolrawbox\the\l@dpscL}%
          958 {\vbadness=10000
          959 \splittopskip=\z@
          960 \do@lineLhook
          961 \l@demtyd@ta
          962 \global\setbox\one@line=\vsplit\namebox{\l@dLcolrawbox\the\l@dpscL}
          963 to\baselineskip}%
          964 \IfStrEq{\splitfirstmarks\parledgroup@}{\begin}{\parledgroup@notes@startL}{\}
          965 \unvbox\one@line \global\setbox\one@line=\lastbox
          966 \getline@numL
          967 \ifnum\@lock>\@ne%
          968 \inserthangingsymboltrue%
          969 \else%
          970 \inserthangingsymbolfalse%
          971 \fi
          972 \setbox\l@dleftbox
          973 \hb@xt@ \Lcolwidth{%
          974 \affixline@num
          975 \add@inserts
          976 \affixside@note
          977 \xifinlist{\the\l@dpscL}{\eled@sections@@}%
          978 {}%
          979 {\print@lineL}}%
          980 \add@penaltiesL
          981 \global\advance\@donereallinesL\@ne
          982 \global\advance\@donetotallinesL\@ne
          983 \else
          984 \setbox\l@dleftbox \hb@xt@ \Lcolwidth{\hspace*\Lcolwidth}}%

```

```

985 \global\advance\@donetotallinesL\@ne
986 \fi}
987
988

```

\print@eledsectionL \print@lineL is for lines without a sectioning command.

```

989 \def\print@lineL{%
990   \affixpstart@numL%
991   \l@dld@ta %space kept for backwar compatibility
992   \l@dlsn@te %space kept for backwar compatibility
993   {\ledllfill\hb@xt@ \wd\one@line{\do@insidelineLhook\inserthangingsymbolL\new@lineL\l@dunhbox@line
994     \l@drsn@te}}

```

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

```

995 \def\print@eledsectionL{%
996   \addtocounter{pstartL}{-1}%
997   \ifdefstring{\@eledsectnotoc}{L}{\ledsectnotoc}{%}
998   \ifdefstring{\@eledsectmark}{L}{\ledsectnomark}{%}
999   \numdef{temp@}{\l@dpscL-1}%
1000  \xifinlist{temp@}{\eled@sections@}{\@nobreaktrue}{\@nobreakfalse}%
1001  \@eled@sectioningtrue%
1002  \csuse{eled@sectioning@the\l@dpscL}%
1003  \@eled@sectioningfalse%
1004  \global\csundef{eled@sectioning@the\l@dpscL}%
1005  \if@RTL%
1006    \hspace{-\paperwidth}%
1007    {\hbox{\l@dunhbox@line{\one@line}} \new@line}%
1008  \else%
1009    \hspace{\paperwidth}%
1010    {\new@line \hbox{\l@dunhbox@line{\one@line}}}%
1011  \fi%
1012  \vskip\eledsection@correcting@skip%
1013 }

```

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

```

\do@lineRhook 1014 \newcommand*{\do@lineLhook}{%}
\do@insidelineLhook 1015 \newcommand*{\do@lineRhook}{%}
\do@insidelineRhook 1016 \newcommand*{\do@insidelineLhook}{%}
1017 \newcommand*{\do@insidelineRhook}{%}
1018

```

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

```

1019 \newcommand*{\do@lineR}{%
1020   \ledRcol@true%
1021   \advance\countRline \@ne
1022   \ifvbox\namebox{\l@dRcolrawbox\the\l@dpscR}%
1023   {\vbadness=10000

```

```

1024 \splittopskip=\z@
1025 \do@lineRhook
1026 \l@emptyd@ta
1027 \global\setbox\one@lineR=\vsplit\namebox{l@dRcolrawbox\the\l@dpscR}
1028         to\baselineskip}%
1029 \IfStrEq{\splitfirstmarks\parledgroup@}{begin}{\parledgroup@notes@startR}{\}
1030 \unvbox\one@lineR \global\setbox\one@lineR=\lastbox
1031 \getline@numR
1032 \ifnum\@lockR>\@ne%
1033     \inserthangingsymbolRtrue
1034 \else%
1035     \inserthangingsymbolRfalse%
1036 \fi%
1037 \setbox\l@dtrightbox
1038 \hb@xt@ \Rcolwidth{%
1039     \affixline@numR%
1040     \add@insertsR
1041     \affixside@noteR
1042     \xifinlist{\the\l@dpscR}{\eled@sectionsR@@}%
1043     }%
1044     {\print@lineR}%
1045 }%
1046 \add@penaltiesR
1047 \global\advance\@donereallinesR\@ne
1048 \global\advance\@donetotallinesR\@ne
1049 \else
1050 \setbox\l@dtrightbox \hb@xt@ \Rcolwidth{\hspace*{\Rcolwidth}}
1051 \global\advance\@donetotallinesR\@ne
1052 \fi
1053 \ledRcol@false%
1054 }
1055
1056

```

```

\print@lineR
\print@eledsectionR

```

### 17.3 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.

```

1057 \newcommand*{\getline@numR}{%
1058     \global\advance\absline@numR \@ne
1059     \do@actionsR
1060     \do@ballastR
1061 \ifledgroupnotesR\else\ifnumberline
1062     \ifsublines@
1063         \ifnum\sub@lockR<\tw@
1064             \global\advance\subline@numR \@ne
1065         \fi

```

```

1066 \else
1067   \ifnum\@lockR<\tw@
1068     \global\advance\line@numR \@ne
1069     \global\subline@numR \z@
1070   \fi
1071 \fi
1072 \fi
1073 \fi
1074 }
1075 \newcommand*\getline@numL{%
1076   \global\advance\absline@num \@ne
1077   \do@actions
1078   \do@ballast
1079   \ifledgroupnotesL@\else\ifnumberline
1080     \ifsublines@
1081       \ifnum\sub@lock<\tw@
1082         \global\advance\subline@num \@ne
1083       \fi
1084     \else
1085       \ifnum\@lock<\tw@
1086         \global\advance\line@num \@ne
1087         \global\subline@num \z@
1088       \fi
1089     \fi
1090 \fi
1091 \fi
1092 }
1093
1094

```

`\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.

```

1095 \newcommand*\do@ballastR{\global\ballast@count=\z@
1096   \begingroup
1097     \advance\absline@numR \@ne
1098     \ifnum\next@actionlineR=\absline@numR
1099       \ifnum\next@actionR>-1001
1100         \global\advance\ballast@count by -\c@ballast
1101       \fi
1102     \fi
1103   \endgroup}

```

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

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

```

1104 \newcommand*\do@actions@fixedcodeR{%
1105   \ifcase\@l@dttempcnta%

```

```

1106 \or% % 1001
1107 \global\sublines@true
1108 \or% % 1002
1109 \global\sublines@false
1110 \or% % 1003
1111 \global\@lockR=\@ne
1112 \or% % 1004
1113 \ifnum\@lockR=\tw@
1114 \global\@lockR=\thr@@
1115 \else
1116 \global\@lockR=\z@
1117 \fi
1118 \or% % 1005
1119 \global\sub@lockR=\@ne
1120 \or% % 1006
1121 \ifnum\sub@lockR=\tw@
1122 \global\sub@lockR=\thr@@
1123 \else
1124 \global\sub@lockR=\z@
1125 \fi
1126 \or% % 1007
1127 \l@dskipnumbertrue
1128 \else
1129 \led@warn@BadAction
1130 \fi}
1131
1132
1133 \newcommand*{\do@actionsR}{%
1134 \global\let\do@actions@nextR=\relax
1135 \@l@dttempcntb=\absline@numR
1136 \ifnum\@l@dttempcntb<\next@actionlineR\else
1137 \ifnum\next@actionR>-1001\relax
1138 \global\page@numR=\next@actionR
1139 \ifbypage@R
1140 \global\line@numR \z@ \global\subline@numR \z@
1141 \fi
1142 \else
1143 \ifnum\next@actionR<-4999\relax % 9/05 added relax here
1144 \@l@dttempcnta=-\next@actionR
1145 \advance\@l@dttempcnta by -5001\relax
1146 \ifsublines@
1147 \global\subline@numR=\@l@dttempcnta
1148 \else
1149 \global\line@numR=\@l@dttempcnta
1150 \fi
1151 \else
1152 \@l@dttempcnta=-\next@actionR
1153 \advance\@l@dttempcnta by -1000\relax
1154 \do@actions@fixedcodeR
1155 \fi

```

```

1156 \fi
1157 \ifx\actionlines@listR\empty
1158 \gdef\next@actionlineR{1000000}%
1159 \else
1160 \gl@p\actionlines@listR\to\next@actionlineR
1161 \gl@p\actions@listR\to\next@actionR
1162 \global\let\do@actions@nextR=\do@actionsR
1163 \fi
1164 \fi
1165 \do@actions@nextR}
1166

```

## 17.4 Line number printing

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

```

\ch@cksub@l@ckR 1167
\ch@ck@l@ckR 1168 \providecommand*\l@dcalcnun}[3]{%
\fx@l@cksR 1169 \ifnum #1 > #2\relax
\affixline@numR 1170 \@l@tempcnta = #1\relax
1171 \advance\@l@tempcnta by -#2\relax
1172 \divide\@l@tempcnta by #3\relax
1173 \multiply\@l@tempcnta by #3\relax
1174 \advance\@l@tempcnta by #2\relax
1175 \else
1176 \@l@tempcnta=#2\relax
1177 \fi}
1178
1179 \newcommand*\ch@cksub@l@ckR}{%
1180 \ifcase\sub@lockR
1181 \or
1182 \ifnum\sublock@disp=\@ne
1183 \@l@tempcntb \z@ \@l@tempcnta \@ne
1184 \fi
1185 \or
1186 \ifnum\sublock@disp=\tw@
1187 \else
1188 \@l@tempcntb \z@ \@l@tempcnta \@ne
1189 \fi
1190 \or
1191 \ifnum\sublock@disp=\z@
1192 \@l@tempcntb \z@ \@l@tempcnta \@ne
1193 \fi
1194 \fi}
1195
1196 \newcommand*\ch@ck@l@ckR}{%
1197 \ifcase\@lockR
1198 \or
1199 \ifnum\lock@disp=\@ne
1200 \@l@tempcntb \z@ \@l@tempcnta \@ne

```

```

1201 \fi
1202 \or
1203 \ifnum\lock@disp=\tw@
1204 \else
1205 \l@dttempcntb \z@ \l@dttempcnta \@ne
1206 \fi
1207 \or
1208 \ifnum\lock@disp=\z@
1209 \l@dttempcntb \z@ \l@dttempcnta \@ne
1210 \fi
1211 \fi}
1212
1213 \newcommand*{\f@x@l@cksR}{%
1214 \ifcase\@lockR
1215 \or
1216 \global\@lockR \tw@
1217 \or \or
1218 \global\@lockR \z@
1219 \fi
1220 \ifcase\sub@lockR
1221 \or
1222 \global\sub@lockR \tw@
1223 \or \or
1224 \global\sub@lockR \z@
1225 \fi}
1226
1227
1228 \newcommand*{\affixline@numR}{%
1229 \ifledgroupnotesR\else\ifnumberline
1230 \ifl@dskipnumber
1231 \global\l@dskipnumberfalse
1232 \else
1233 \ifsublines@
1234 \l@dttempcntb=\subline@numR
1235 \l@dcalcnnum{\subline@numR}{\c@firstsublinenumR}{\c@sublinenumincrementR}%
1236 \ch@cksub@lockR
1237 \else
1238 \l@dttempcntb=\line@numR
1239 \ifx\linenumberlist\empty
1240 \l@dcalcnnum{\line@numR}{\c@firstlinenumR}{\c@linenumincrementR}%
1241 \else
1242 \l@dttempcnta=\line@numR
1243 \edef\rem@inder{\linenumberlist,\number\line@numR,%
1244 \edef\sc@n@list{\def\noexpand\sc@n@list
1245 ###1,\number\l@dttempcnta,###2|{\def\noexpand\rem@inder{###2}}}%
1246 \sc@n@list\expandafter\sc@n@list\rem@inder|}%
1247 \ifx\rem@inder\empty\advance\l@dttempcnta\@ne\fi
1248 \fi
1249 \ch@ck@l@ckR
1250 \fi

```



```

1251 \ifnum\@l@tempcnta=\@l@tempcntb
1252   \if@twocolumn
1253     \if@firstcolumn
1254       \gdef\l@dld@ta{\llap{\leftlinenumR}}}%
1255     \else
1256       \gdef\l@drd@ta{\rlap{\rightlinenumR}}}%
1257     \fi
1258   \else
1259     \@l@tempcntb=\line@marginR
1260     \ifnum\@l@tempcntb>\@ne
1261       \advance\@l@tempcntb by\page@numR
1262     \fi
1263     \ifodd\@l@tempcntb
1264       \gdef\l@drd@ta{\rlap{\rightlinenumR}}}%
1265     \else
1266       \gdef\l@dld@ta{\llap{\leftlinenumR}}}%
1267     \fi
1268   \fi
1269 \fi
1270 \f@x@l@cksR
1271 \fi
1272 \fi
1273 \fi}

```

## 17.5 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 1274
  \leftpstartnumR 1275 \newcommand*{\affixpstart@numL}{%
\rightpstartnumR 1276 \ifsidepstartnum
  \leftpstartnumL 1277 \if@twocolumn
\rightpstartnumL 1278   \if@firstcolumn
  \ifpstartnumR 1279     \gdef\l@dld@ta{\llap{\leftpstartnumL}}}%
1280   \else
1281     \gdef\l@drd@ta{\rlap{\rightpstartnumL}}}%
1282   \fi
1283   \else
1284     \@l@tempcntb=\line@margin
1285     \ifnum\@l@tempcntb>\@ne
1286       \advance\@l@tempcntb \page@num
1287     \fi
1288     \ifodd\@l@tempcntb

```

```

1289     \gdef\l@drd@ta{\rlap{{\rightpstartnumL}}}%
1290     \else
1291     \gdef\l@dld@ta{\llap{{\leftpstartnumL}}}%
1292     \fi
1293     \fi
1294 \fi
1295 }
1296 \newcommand*{\affixpstart@numR}{%
1297 \ifsidepstartnum
1298 \if@twocolumn
1299     \if@firstcolumn
1300     \gdef\l@dld@ta{\llap{{\leftpstartnumR}}}%
1301     \else
1302     \gdef\l@drd@ta{\rlap{{\rightpstartnumR}}}%
1303     \fi
1304     \else
1305     \@l@tempcntb=\line@marginR
1306     \ifnum\@l@tempcntb>\@ne
1307     \advance\@l@tempcntb \page@numR
1308     \fi
1309     \ifodd\@l@tempcntb
1310     \gdef\l@drd@ta{\rlap{{\rightpstartnumR}}}%
1311     \else
1312     \gdef\l@dld@ta{\llap{{\leftpstartnumR}}}%
1313     \fi
1314     \fi
1315 \fi
1316 }
1317
1318 \newcommand*{\leftpstartnumL}{
1319 \ifpstartnum
1320 \thepstartL
1321 \kern\linenumsep\global\pstartnumfalse\fi
1322 }
1323 \newcommand*{\rightpstartnumL}{
1324 \ifpstartnum\kern\linenumsep
1325 \thepstartL
1326 \global\pstartnumfalse\fi
1327 }
1328 \newif\ifpstartnumR
1329 \pstartnumRtrue
1330 \newcommand*{\leftpstartnumR}{
1331 \ifpstartnumR
1332 \thepstartR
1333 \kern\linenumsep\global\pstartnumRfalse\fi
1334 }
1335 \newcommand*{\rightpstartnumR}{
1336 \ifpstartnumR\kern\linenumsep
1337 \thepstartR
1338 \global\pstartnumRfalse\fi

```

1339 }

## 17.6 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.

1340 `\list@create{\inserts@listR}`

`\add@insertsR` The right text version.

```
\add@inserts@nextR 1341 \newcommand*{\add@insertsR}{%
1342   \global\let\add@inserts@nextR=\relax
1343   \ifx\inserts@listR\empty \else
1344     \ifx\next@insertR\empty
1345       \ifx\insertlines@listR\empty
1346         \global\noteschanged@true
1347         \gdef\next@insertR{100000}%
1348       \else
1349         \gl@p\insertlines@listR\to\next@insertR
1350       \fi
1351     \fi
1352     \ifnum\next@insertR=\absline@numR
1353       \gl@p\inserts@listR\to\@insertR
1354       \@insertR
1355       \global\let\@insertR=\undefined
1356       \global\let\next@insertR=\empty
1357       \global\let\add@inserts@nextR=\add@insertsR
1358     \fi
1359   \fi
1360   \add@inserts@nextR}
1361
```

## 17.7 Penalties

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

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

```
\newcommand*{\add@penaltiesR}{\@l@dttempcnta=\ballast@count
\ifnum\num@linesR>\@ne
\global\advance\par@lineR \@ne
```

```

\ifnum\par@lineR=\@ne
  \advance\@l@dttempcnta by \clubpenalty
\fi
\@l@dttempcntb=\par@lineR \advance\@l@dttempcntb \@ne
\ifnum\@l@dttempcntb=\num@linesR
  \advance\@l@dttempcnta by \widowpenalty
\fi
\ifnum\par@lineR<\num@linesR
  \advance\@l@dttempcnta by \interlinepenalty
\fi
\fi
\ifnum\@l@dttempcnta=\z@
  \relax
\else
  \ifnum\@l@dttempcnta>-10000
    \penalty\@l@dttempcnta
  \else
    \penalty -10000
  \fi
\fi}

```

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

```

1362 \newcommand*{\add@penaltiesL}{\}
1363 \newcommand*{\add@penaltiesR}{\}
1364

```

## 17.8 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.

```

1365 \newcommand*{\flush@notesR}{\%
1366   \@xloop
1367   \ifx\inserts@listR\empty \else
1368     \gl@p\inserts@listR\to\@insertR
1369     \@insertR
1370     \global\let\@insertR=\undefined
1371   \repeat}
1372

```

## 18 Footnotes

### 18.1 Normal footnote formatting

The `\printlines` macro prints the line numbers for a note—which, in the general case, is a rather complicated task. The seven parameters of the argument are the line numbers as stored in `\l@d@nums`, in the form described on page ??: 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:
\printlinesR      #1      | #2 | #3 | #4 | #5 | #6 | #7
\printlinesR start-page | line | subline | end-page | line | subline | font
1373 \def\printlinesR#1|#2|#3|#4|#5|#6|#7|{\begingroup
1374   \setprintlines{#1}{#2}{#3}{#4}{#5}{#6}%
1375   \ifl@d@pnum #1\fullstop\fi
1376   \ifledplinenum \linenumr@p{#2}\Rlineflag\else \sympninenum\fi
1377   \ifl@d@ssub \fullstop \sublinenumr@p{#3}\fi
1378   \ifl@d@dash \endashchar\fi
1379   \ifl@d@pnum #4\fullstop\fi
1380   \ifl@d@elin \linenumr@p{#5}\Rlineflag\fi
1381   \ifl@d@esl \ifl@d@elin \fullstop\fi \sublinenumr@p{#6}\fi
1382 \endgroup}
1383
1384 \let\ledsavedprintlines\printlines
1385

```

## 19 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.

```

1386 \list@create{\labelref@listR}
1387

```

`\edlabel` The `\edlabel` command first writes a `\@lab` macro to the `\linenum@out` file. It then checks to see that the `\labelref@list` actually has something in it (if not, it creates a dummy entry), and pops the next value for the current label, storing it in `\label@refs`. Finally it defines the label to be `\empty` so that any future check will turn up the fact that it has been used.

```

1388 \renewcommand*{\edlabel}[1]{\@bsphack
1389   \ifledRcol
1390     \write\linenum@outR{\string\@lab}%
1391     \ifx\labelref@listR\empty
1392       \xdef\label@refs{\zz@@@}%
1393     \else

```

```

1394     \gl@p\labelref@listR\to\label@refs
1395     \fi
1396     \ifvmode
1397         \advancelabel@refs
1398     \fi
1399     \protected@write\@auxout{%
1400         {\string\l@dmake@labelsR\space\thepage|\label@refs|\the\c@pstartR|{#1}}%
1401     \else
1402         \write\linenum@out{\string\@lab}%
1403         \ifx\labelref@list\empty
1404             \xdef\label@refs{\zz@@}%
1405         \else
1406             \gl@p\labelref@list\to\label@refs
1407         \fi
1408         \ifvmode
1409             \advancelabel@refs
1410         \fi
1411         \protected@write\@auxout{%
1412             {\string\l@dmake@labelsR\space\thepage|\label@refs|\the\c@pstartR|{#1}}%
1413         \fi
1414         \@esphack}
1415
1416

```

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

```

1417 \def\l@dmake@labelsR#1|#2|#3|#4|#5{%
1418     \expandafter\ifx\csname the@label#5\endcsname \relax\else
1419         \led@warn@DuplicateLabel{#4}%
1420     \fi
1421     \expandafter\gdef\csname the@label#5\endcsname{#1|#2\Rlineflag|#3|#4}%
1422     \ignorespaces}
1423 \AtBeginDocument{%
1424     \def\l@dmake@labelsR#1|#2|#3|#4|#5{%
1425 }
1426

```

`\@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.

```

1427 \renewcommand*{\@lab}{%
1428     \ifledRcol
1429         \xright@appenditem{\linenumr@p{\line@numR}}|{%
1430             \ifsublines@ \sublinenumr@p{\subline@numR}\else 0\fi}%
1431         \to\labelref@listR
1432     \else
1433         \xright@appenditem{\linenumr@p{\line@num}}|{%
1434             \ifsublines@ \sublinenumr@p{\subline@num}\else 0\fi}%
1435         \to\labelref@list

```

```

1436 \fi}
1437

```

## 20 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* 1438 \WithSuffix\newcommand\sidenotemargin*[1]{%
1439 \l@dgetsidenote@margin{#1}
1440 \global\sidenote@marginR=\@l@dttempcntb
1441 \global\sidenote@margin=\@l@dttempcntb
1442 }
1443 \newcount\sidenote@marginR
1444 \global\sidenote@margin=\@ne
1445

```

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

```

1446 \newcommand*\affixside@noteR{%
1447 \def\sidenotecontent@{}%
1448 \numdef\itemcount@{0}%
1449 \def\do##1{%
1450 \ifnumequal{\itemcount@}{0}%
1451 {%
1452 \appto\sidenotecontent@{##1}}% Not print not separator before the 1st note
1453 {\appto\sidenotecontent@{\sidenotesep ##1}}%
1454 }%
1455 \numdef\itemcount@{\itemcount@+1}%
1456 }%
1457 \dolistloop{\l@dcnotesetext}%
1458 \ifnumgreater{\itemcount@}{1}{\eledmac@warning{\itemcount@\space sidenotes on line \the\line@numR}}
1459 \gdef\@templ@d{%
1460 \ifx\@templ@d\l@dcnotesetext \else%
1461 \if@twocolumn%
1462 \if@firstcolumn%
1463 \setl@dlp@rbox{##1}{\sidenotecontent@}%
1464 \else%
1465 \setl@drp@rbox{\sidenotecontent@}%
1466 \fi%
1467 \else%
1468 \l@dttempcntb=\sidenote@marginR%
1469 \ifnum\l@dttempcntb>\@ne%
1470 \advance\l@dttempcntb by\page@numR%
1471 \fi%
1472 \ifodd\l@dttempcntb%
1473 \setl@drp@rbox{\sidenotecontent@}%
1474 \gdef\sidenotecontent@{}%

```

```

1475     \numdef{\itemcount@}{0}%
1476     \dolistloop{\l@dcstotetext@l}%
1477     \ifnumgreater{\itemcount@}{1}{\eledmac@warning{\itemcount@\space leftnotes on line
1478     \setl@dlp@rbox{\sidenotecontent@}%
1479     \else%
1480     \setl@dlp@rbox{\sidenotecontent@}%
1481     \gdef\sidenotecontent@{}%
1482     \numdef{\itemcount@}{0}%
1483     \dolistloop{\l@dcstotetext@r}%
1484     \ifnumgreater{\itemcount@}{1}{\eledmac@warning{\itemcount@\space rightnotes on line
1485     \setl@drp@rbox{\sidenotecontent@}%
1486     \fi%
1487     \fi%
1488     \fi%
1489 }
1490

```

## 21 Familiar footnotes

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

```

1491 \renewcommand{\l@dbfnote}[1]{%
1492   \ifnumberedpar@
1493   \gdef\@tag{#1}%
1494   \ifledRcol%
1495     \xright@appenditem{\noexpand\vl@dbfnote{\csexpandonce{\@tag}}{\@thefnmark}}%
1496     \to\inserts@listR
1497     \global\advance\insert@countR \@ne%
1498   \else%
1499     \xright@appenditem{\noexpand\vl@dbfnote{\csexpandonce{\@tag}}{\@thefnmark}}%
1500     \to\inserts@list
1501     \global\advance\insert@count \@ne%
1502   \fi
1503   \fi\ignorespaces}
1504

```

`\normalbfnoteX`

```

1505 \renewcommand{\normalbfnoteX}[2]{%
1506   \ifnumberedpar@
1507   \ifledRcol%
1508   \ifluatex
1509     \footnotelang@lua[R]%
1510   \fi
1511   \ifundefined{xpg@main@language}%if polyglossia
1512     {}%
1513     {\footnotelang@poly[R]}%
1514   \protected@csxdef{thisfootnote}{\csuse{thefootnote#1}}%
1515   \xright@appenditem{\noexpand\vbfnoteX{#1}{#2}{\csexpandonce{thisfootnote}}}%

```



```

1516             \to\inserts@listR
1517     \global\advance\insert@countR \@ne%
1518 \else%
1519     \ifluatex
1520         \footnotelang@lua%
1521     \fi
1522     \@ifundefined{xpg@main@language}%if polyglossia
1523     {}%
1524     {\footnotelang@poly}%
1525     \protected@csxdef{thisfootnote}{\csuse{thefootnote#1}}%
1526     \xright@appenditem{\noexpand\vbfnoteX{#1}{#2}}{\csexpandonce{thisfootnote}}}%
1527     \to\inserts@list
1528     \global\advance\insert@count \@ne%
1529 \fi
1530 \fi\ignorespaces}
1531

```

## 22 Verse

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

```

\inserthangingsymbolL
\inserthangingsymbolR 1532 \newif\ifinserthangingsymbolR
1533 \newcommand{\inserthangingsymbolL}{%
1534 \ifinserthangingsymbol%
1535     \ifinstanzaL%
1536         \hangingsymbol%
1537     \fi%
1538 \fi}
1539 \newcommand{\inserthangingsymbolR}{%
1540 \ifinserthangingsymbolR%
1541     \ifinstanzaR%
1542         \hangingsymbol%
1543     \fi%
1544 \fi}

```

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

```

\correcthangingL
\correcthangingR 1545 \newcommand{\correcthangingL}{%
1546 \ifl@dpageing\else%
1547     \ifinstanzaL%
1548         \ifinserthangingsymbol%
1549             \hskip \@ifundefined{sza@0@}{0}{\expandafter%
1550                 \noexpand\csname sza@0@endcsname}\stanzaindentbase%

```

```

1551         \fi%
1552     \fi%
1553 \fi}
1554
1555 \newcommand{\correcthangingR}{%
1556 \ifl@dpaging\else%
1557     \ifinstanzaR%
1558         \ifinserthangingsymbolR%
1559             \hskip \@ifundefined{sza@00}{0}{\expandafter%
1560                 \noexpand\csname sza@00\endcsname}\stanzaindentbase%
1561         \fi%
1562     \fi%
1563 \fi}

```

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

```

1564 \chardef\next=\catcode'\&
1565 \catcode'\&=\active
1566

```

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

```

1567 \newenvironment{astanza}{%
1568     \startstanzahook
1569     \catcode'\&\active
1570     \global\stanza@count\@ne\stanza@modulo\@ne
1571     \ifnum\usernamecount{sza@00}=\z@
1572         \let\stanza@hang\relax
1573         \let\endlock\relax
1574     \else
1575     %% \interlinepenalty\@M % this screws things up, but I don't know why
1576         \rightskip\z@ plus 1fil\relax
1577     \fi
1578     \ifnum\usernamecount{szp@00}=\z@
1579         \let\sza@penalty\relax
1580     \fi
1581     \def&{%
1582         \endlock\mbox{}%
1583         \sza@penalty
1584         \global\advance\stanza@count\@ne
1585         \@astanza@line}%
1586     \def\&{%
1587         \endlock\mbox{}
1588         \pend
1589         \endstanzaextra}%
1590     \pstart
1591     \@astanza@line
1592 }{}
1593

```

`\@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.

```

1594 \newcommand*{\@astanza@line}{%
1595   \ifnum\value{stanzaindentrepetition}=0
1596     \parindent=\csname sza@\number\stanza@count
1597               @\endcsname\stanzaindentbase
1598   \else
1599     \parindent=\csname sza@\number\stanza@modulo
1600               @\endcsname\stanzaindentbase
1601     \managestanza@modulo
1602   \fi
1603   \par
1604   \stanza@hang%\mbox{}%
1605   \ignorespaces}
1606

```

Lastly reset the modified category codes.

```

1607 \catcode'\&=\next
1608

```

## 23 Naming macros

The LaTeX 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 1609 \providecommand*{\newnamebox}[1]{%
\unvnamebox 1610 \expandafter\newbox\csname #1\endcsname}
\namebox 1611 \providecommand*{\setnamebox}[1]{%
1612   \expandafter\setbox\csname #1\endcsname}
1613 \providecommand*{\unhnamebox}[1]{%
1614   \expandafter\unhbox\csname #1\endcsname}
1615 \providecommand*{\unvnamebox}[1]{%
1616   \expandafter\unvbox\csname #1\endcsname}
1617 \providecommand*{\namebox}[1]{%
1618   \csname #1\endcsname}
1619

```

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

```

\usenamecount 1620 \providecommand*{\newnamecount}[1]{%
1621   \expandafter\newcount\csname #1\endcsname}
1622 \providecommand*{\usenamecount}[1]{%
1623   \csname #1\endcsname}
1624

```

## 24 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.
1625 \newcount\l@dc@maxchunks
1626 \newcommand{\maxchunks}[1]{\l@dc@maxchunks=#1}
1627 \maxchunks{5120}
1628

\l@dnumpstartsL The numbers of left and right chunks. \l@dnumpstartsL is defined in eledmac.
\l@dnumpstartsR 1629 \newcount\l@dnumpstartsR
1630

\l@pscL A couple of scratch counts for use in left and right texts, respectively.
\l@pscR 1631 \newcount\l@dpscL
1632 \newcount\l@dpscR
1633

\l@dssetuprawboxes This macro creates \maxchunks pairs of boxes for left and right chunks. The boxes
are called \l@dLcolrawbox1, \l@dLcolrawbox2, etc.
1634 \newcommand*{\l@dssetuprawboxes}{%
1635 \l@l@tempcntb=\l@dc@maxchunks
1636 \loop\ifnum\l@l@tempcntb>\z@
1637 \newnamebox{\l@dLcolrawbox\the\l@l@tempcntb}
1638 \newnamebox{\l@dRcolrawbox\the\l@l@tempcntb}
1639 \advance\l@l@tempcntb \m@ne
1640 \repeat}
1641

\l@dssetupmaxlinecounts 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@dssetupmaxlinecounts creates
\maxchunks new counts called \l@dmaxlinesinpar1, etc., and \l@dzeromaxlinecounts
zeroes all of them.
1642 \newcommand*{\l@dssetupmaxlinecounts}{%
1643 \l@l@tempcntb=\l@dc@maxchunks
1644 \loop\ifnum\l@l@tempcntb>\z@
1645 \newnamecount{\l@dmaxlinesinpar\the\l@l@tempcntb}
1646 \advance\l@l@tempcntb \m@ne
1647 \repeat}
1648 \newcommand*{\l@dzeromaxlinecounts}{%
1649 \begingroup
1650 \l@l@tempcntb=\l@dc@maxchunks
1651 \loop\ifnum\l@l@tempcntb>\z@

```

```

1652   \global\usernamecount{l@maxlinesinpar\the\l@dttempcntb}=\z@
1653   \advance\l@dttempcntb \m@ne
1654   \repeat
1655   \endgroup}
1656

```

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

```

1657 \AtBeginDocument{%
1658   \l@dsetuprawboxes
1659   \l@dsetupmaxlinecounts
1660   \l@dzeromaxlinecounts
1661   \l@dnumstartsl=\z@
1662   \l@dnumstartsr=\z@
1663   \l@dpscl=\z@
1664   \l@dpscr=\z@}
1665

```

## 25 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 1666 \newif\ifl@dusedbabel
\l@dusedbabeltrue 1667 \l@dusedbabelfalse

\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{\langle lang\rangle} does the work when the language
\langle lang\rangle 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 \.
1668 \newcommand*{\l@dbbl@set@language}[1]{%
1669   \edef\language{#1}%
1670   \select@language{\language}%

```

```

1671 \if@filesw
1672   \protected@write\@auxout{}\string\select@language{\language}%
1673   \addtocontents{toc}{\string\select@language{\language}}%
1674   \addtocontents{lof}{\string\select@language{\language}}%
1675   \addtocontents{lot}{\string\select@language{\language}}%
1676 \fi}
1677

```

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
1678 \providecommand{\selectlanguage}[1]{%
1679 \newcommand*\l@duselanguage[1]{%
1680 \gdef\theledlanguageL{}
1681 \gdef\theledlanguageR{}
1682

```

Now do the `babel` fix or `polyglossia`, if necessary.

```

1683 \AtBeginDocument{%
1684   \ifundefined{xpg@main@language}{%
1685     \ifundefined{bbl@main@language}{%

```

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

```

1686   \l@dusedbabelfalse
1687   \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.

```

1688   \l@dusedbabeltrue
1689   \let\l@doldselectlanguage\selectlanguage
1690   \let\l@doldbbl@set@language\bbl@set@language
1691   \let\bbl@set@language\l@dbbl@set@language
1692   \renewcommand{\selectlanguage}[1]{%
1693     \l@doldselectlanguage{#1}%
1694     \ifledRcol \gdef\theledlanguageR{#1}%
1695     \else      \gdef\theledlanguageL{#1}%
1696   \fi}

```

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

```

1697   \renewcommand*\l@duselanguage[1]{%
1698     \l@doldselectlanguage{#1}}

```

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

```

1699   \gdef\theledlanguageL{\bbl@main@language}%
1700   \gdef\theledlanguageR{\bbl@main@language}%

```

```

1701 }%
1702 }
    If on Polyglossia
1703 { \let\old@otherlanguage\otherlanguage%
1704   \renewcommand{\otherlanguage}[2] [] {%
1705     \selectlanguage[#1]{#2}%
1706     \ifledRcol \gdef\theledlanguageR{#2}%
1707     \else      \gdef\theledlanguageL{#2}%
1708     \fi}%
1709   \let\l@duselanguage\xpg@set@language%
1710   \gdef\theledlanguageL{\xpg@main@language}%
1711   \gdef\theledlanguageR{\xpg@main@language}%
1712 % \end{macrocode}
1713 % That's it.
1714 % \begin{macrocode}
1715 }}

```

## 26 Parallel columns

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

```

\@eledsectionR 1716 \newsavebox{\@eledsectionL}%
1717 \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.

```

1718 \newcommand*{\Columns}{%
1719   \eledsection@correcting@skip=-\baselineskip% Correction for sections' titles
1720   \setcounter{pstartL}{\value{pstartLold}}
1721   \setcounter{pstartR}{\value{pstartRold}}
1722   \ifnum\l@dnumpstartsL=\l@dnumpstartsR\else
1723     \led@err@BadLeftRightPstarts{\the\l@dnumpstartsL}{\the\l@dnumpstartsR}%
1724   \fi

```

Start a group and zero counters, etc.

```

1725 \begingroup
1726   \l@dzeropenalties
1727   \endgraf\global\num@lines=\prevgraf
1728   \global\num@linesR=\prevgraf
1729   \global\par@line=\z@
1730   \global\par@lineR=\z@
1731   \global\l@dpscL=\z@
1732   \global\l@dpscR=\z@

```

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

```

1733 \check@pstarts
1734 \loop\if@pstarts

```

```

1735      \global\pstartnumtrue
1736      \global\pstartnumRtrue

Increment \l@dpscL and \l@dpscR which here count the numbers of left and right
chunks.

1737      \global\advance\l@dpscL \@ne
1738      \global\advance\l@dpscR \@ne

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

1739      \checkraw@text
1740 {      \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.

1741      \l@duselanguage{\theledlanguageL}%
1742      \do@lineL
1743      \xifinlist{\the\l@dpscL}{\eled@sections@@}
1744      {%
1745      \ifdefstring{\@eledsectmark}{L}%
1746      {\csuse{eled@sectmark@the\l@dpscL}%
1747      }-}%
1748      \global\csundef{eled@sectmark@the\l@dpscL}%
1749      \savebox{\@eledsectionL}{\parbox[t][t]{\Lcolwidth}{\vbox{} \print@eledse
1750      }%
1751      }%
1752      \l@duselanguage{\theledlanguageR}%
1753      \do@lineR
1754      \xifinlist{\the\l@dpscR}{\eled@sectionsR@@}
1755      {%
1756      \ifdefstring{\@eledsectmark}{R}%
1757      {\csuse{eled@sectmark@the\l@dpscR R}%
1758      }-}%
1759      \global\csundef{eled@sectmark@the\l@dpscR R}%
1760      \savebox{\@eledsectionR}{\parbox[t][t]{\Rcolwidth}{\vbox{} \print@eledse
1761      }%
1762      \hb@xt@ \hsize{%
1763      \ifdefstring{\columns@position}{L}{-}{\hfill }%
1764      \unhbox\l@dleftbox%
1765      \ifhbox\@eledsectionL%
1766      \usebox{\@eledsectionL}%
1767      \fi%
1768      \print@columnseparator%
1769      \unhbox\l@drightbox%
1770      \ifhbox\@eledsectionR%
1771      \usebox{\@eledsectionR}%
1772      \fi%
1773      \ifdefstring{\columns@position}{R}{-}{\hfill}%
1774      }%
1775      \checkraw@text
1776      \checkverseL

```



```

1777      \checkverseR
1778      \checkpb@columns
1779      \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.

```

1780      \@writelinesinparL
1781      \@writelinesinparR
1782      \check@pstarts
1783      \ifbypstart@
1784      \write\linenum@out{\string\@set[1]}
1785      \resetprevline@
1786      \fi
1787      \ifbypstart@R
1788      \write\linenum@outR{\string\@set[1]}
1789      \resetprevline@
1790      \fi
1791      \addtocounter{pstartL}{1}
1792      \addtocounter{pstartR}{1}
1793      \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.

```

1794      \flush@notes
1795      \flush@notesR
1796      \endgroup
1797      \global\l@dpscL=\z@
1798      \global\l@dpscR=\z@
1799      \global\l@dnumpstartsL=\z@
1800      \global\l@dnumpstartsR=\z@
1801      \ignorespaces
1802      \global\instanzaLfalse
1803      \global\instanzaRfalse}
1804

```

`\print@columnseparator` `\print@columnseparator` print the colum separator, with spaces around, following user's setting. We use the  $\TeX$  `\ifdim` instead of `etoolbox` to avoid having `\hfill` in a `{}`, which delete some (little) space.

```

1805 \def\print@columnseparator{%
1806   \ifdim\beforecolumnseparator<0pt%
1807     \hfill%
1808   \else%
1809     \hspace{\beforecolumnseparator}%
1810   \fi%
1811   \columnseparator%
1812   \ifdim\aftercolumnseparator<0pt%
1813     \hfill%
1814   \else%
1815     \hspace{\beforecolumnseparator}%

```

```

1816 \fi%
1817 }%
1818 %\end{macrocode}
1819 % \end{macro}
1820 % \begin{macro}{\checkpb@columns}
1821 % \cs{checkpb@columns} prevent or make pagebreaking in columns, depending of the use of \
1822 % \begin{macrocode}
1823
1824 \newcommand{\checkpb@columns}{%
1825     \newif\if@pb
1826     \newif\if@nopb
1827     \IfStrEq{\led@pb@setting}{before}{
1828         \numdef{\next@absline}{\the\absline@num+1}%
1829         \numdef{\next@abslineR}{\the\absline@numR+1}%
1830         \xifinlistcs{\next@absline}{l@prev@pb}{\@pbtrue}{}%
1831         \xifinlistcs{\next@abslineR}{l@prev@pbR}{\@pbtrue}{%
1832             \xifinlistcs{\next@absline}{l@prev@nopb}{\@nopbtrue}{}%
1833             \xifinlistcs{\next@abslineR}{l@prev@nopbR}{\@nopbtrue}{%
1834                 }{}
1835             \IfStrEq{\led@pb@setting}{after}{
1836                 \xifinlistcs{\the\absline@num}{l@prev@pb}{\@pbtrue}{}%
1837                 \xifinlistcs{\the\absline@numR}{l@prev@pbR}{\@pbtrue}{%
1838                     \xifinlistcs{\the\absline@num}{l@prev@nopb}{\@nopbtrue}{}%
1839                     \xifinlistcs{\the\absline@numR}{l@prev@nopbR}{\@nopbtrue}{%
1840                         }{}
1841                     \if@nopb\nopagebreak[4]\enlargethispage{\baselineskip}\fi
1842                     \if@pb\pagebreak[4]\fi
1843                 }

```

**\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).

```

1844 \newcommand*{\columnseparator}{%
1845     \smash{\rule[-0.2\baselineskip]{\columnrulewidth}{1.05\baselineskip}}}
1846 \newdimen\columnrulewidth
1847 \columnrulewidth=\z@
1848

```

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

```

1849 \newcommand*{\columnspostion}[1]{%
1850     \xdef\columns@position{#1}%
1851     }%
1852 \xdef\columns@position{R}%

```

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

```

1853 \newlength{\beforecolumnseparator}%
1854 \setlength{\beforecolumnseparator}{-2pt}%
1855
1856 \newlength{\aftercolumnseparator}%
1857 \setlength{\aftercolumnseparator}{-2pt}%

```

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

```

\@pstartstrue 1858 \newif\if@pstarts
\@pstartsfalse 1859 \newcommand*\check@pstarts}{%
\check@pstarts 1860 \@pstartsfalse
1861 \ifnum\l@dnumpstartsL>\l@dpscL
1862 \@pstartstrue
1863 \else
1864 \ifnum\l@dnumpstartsR>\l@dpscR
1865 \@pstartstrue
1866 \fi
1867 \fi
1868 }
1869

```

\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 1870 \newif\ifaraw@text
1871 \araw@textfalse
1872 \newcommand*\checkraw@text}{%
1873 \araw@textfalse
1874 \ifvbox\namebox{\l@dLcolrawbox\the\l@dpscL}
1875 \araw@texttrue
1876 \else
1877 \ifvbox\namebox{\l@dRcolrawbox\the\l@dpscR}
1878 \araw@texttrue
1879 \fi
1880 \fi
1881 }
1882

```

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

```

1883 \newcommand*\@writelinesinparL}{%
1884 \edef\next{%
1885 \write\linenum@out{\string\@pend[\the\@donereallinesL]}}%
1886 \next
1887 \global\@donereallinesL \z@
1888 \newcommand*\@writelinesinparR}{%
1889 \edef\next{%
1890 \write\linenum@outR{\string\@pendR[\the\@donereallinesR]}}%
1891 \next
1892 \global\@donereallinesR \z@
1893

```

## 27 Parallel pages

This is considerably more complicated than parallel columns.

```

\l@minpagelines \numpagelinesL Counts for the number of lines on a left or right page, and the smaller of the
\l@minpagelines \numpagelinesR number of lines on a pair of facing pages.
1894 \newcount\l@minpagelinesL
1895 \newcount\l@minpagelinesR
1896 \newcount\l@minpagelines
1897

\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.

1898 \newcommand*\Pages{%
1899   \eledsection@correcting@skip=-2\baselineskip% line correcting for section titles.
1900   \setcounter{pstartL}{\value{pstartLold}}
1901   \setcounter{pstartR}{\value{pstartRold}}
1902   \parledgroup@notespacing@set@correction
1903   \typeout{}
1904   \typeout{***** PAGES *****}
1905   \ifnum\l@dnumstartsL=\l@dnumstartsR\else
1906     \led@err@BadLeftRightPstarts{\the\l@dnumstartsL}{\the\l@dnumstartsR}%
1907   \fi

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

1908   \cleartol@devenpage
1909   \begingroup
1910     \l@dzeropenalties
1911     \endgraf\global\num@lines=\prevgraf
1912     \global\num@linesR=\prevgraf
1913     \global\par@line=\z@
1914     \global\par@lineR=\z@
1915     \global\l@dpscL=\z@
1916     \global\l@dpscR=\z@
1917     \writtenlinesLfalse
1918     \writtenlinesRfalse

    Check if there are chunks to be processed.

1919     \check@pstarts
1920     \loop\if@pstarts

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

1921       \global\advance\l@dpscL \@ne
1922       \global\advance\l@dpscR \@ne

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

1923       \getlinesfromparlistL

```

```

1924      \getlinesfromparlistR
1925      \l@dcalc@maxoftwo{\@cs@linesinparL}{\@cs@linesinparR}%
1926      {\usernamecount{l@dmaxlinesinpar\the\l@dpscL}}%
1927      \check@pstarts
1928      \repeat

```

Zero the counts again, ready for the next bit.

```

1929      \global\l@dpscL=\z@
1930      \global\l@dpscR=\z@

```

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

```

1931      \getlinesfrompagelistL
1932      \getlinesfrompagelistR
1933      \l@dcalc@minoftwo{\@cs@linesonpageL}{\@cs@linesonpageR}%
1934      {\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.

```

1935      \check@pstarts
1936      \if@pstarts

```

Increment the chunk counts to get the first pair.

```

1937      \global\advance\l@dpscL \@ne
1938      \global\advance\l@dpscR \@ne

```

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

```

1939      \global\@donereallinesL=\z@
1940      \global\@donetotallinesL=\z@
1941      \global\@donereallinesR=\z@
1942      \global\@donetotallinesR=\z@

```

Start a loop over the boxes (chunks).

```

1943      \checkraw@text
1944 %      \begingroup
1945 {      \loop\ifraw@text

```

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

```

1946      \checkpageL
1947      \l@duselanguage{\theledlanguageL}%
1948 %%%      \begingroup
1949 {      \loop\ifl@dsamepage%

```

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

```

1950      \ifdefstring{\@eledsectnotoc}{L}{\ledsectnotoc}{\}
1951      \csuse{before@pstartL@the\l@dpscL}
1952      \global\csundef{before@pstartL@the\l@dpscL}
1953      \do@lineL
1954      \xifinlist{\the\l@dpscL}{\eled@sections@@}

```

```

1955      {\print@eledsectionL}%
1956      {}%
1957      \advance\numpagelinesL \@ne
1958      \ifshiftedpstarts
1959          \ifdim\ht\l@dleftbox>0pt\hb@xt@ \hsize{\ledstrutL\unhbox\l@dleftbox}%
1960      \else%
1961          \parledgroup@correction@notespacing{L}
1962          \hb@xt@ \hsize{\ledstrutL\unhbox\l@dleftbox}%
1963      \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.

```

1964      \get@nextboxL
1965      \checkpageL
1966      \checkverseL
1967      \checkpbL
1968      \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.

```

1969      \ifl@dpagfull
1970          \@writelinesonpageL{\the\numpagelinesL}%
1971      \else
1972          \@writelinesonpageL{1000}%
1973      \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.

```

1974      \numpagelinesL \z@
1975      \parledgroup@correction@notespacing@init
1976      \clearl@dleftpage }%

```

Now do the same for the right text.

```

1977      \checkpageR
1978      \l@duselanguage{\theledlanguageR}%
1979 {
1980      \loop\ifl@dsamepage%
1981          \initnumbering@sectcountR
1982          \ifdefstring{\@eledsectnotoc}{R}{\ledsectnotoc}{%
1983          \csuse{before@pstartR@the\l@dpscR}
1984          \global\csundef{before@pstartR@the\l@dpscR}
1985          \do@lineR
1986          \xifinlist{\the\l@dpscR}{\eled@sectionsR@@}
1987          {\print@eledsectionR}%
1988          {}%
1989      \advance\numpagelinesR \@ne
1990      \ifshiftedpstarts
1991          \ifdim\ht\l@drighbox>0pt\hb@xt@ \hsize{\ledstrutR\unhbox\l@drighbox}%

```

```

1992             \parledgroup@correction@notespacing{R}
1993             \hb@xt@ \hsize{\ledstrutR\unhbox\l@drightbox}%
1994         \fi
1995         \get@nextboxR
1996         \checkpageR
1997         \checkverseR
1998         \checkpbr
1999         \repeat
2000         \ifl@dpagfull
2001             \@writelinesonpageR{\the\numpagelinesR}%
2002         \else
2003             \@writelinesonpageR{1000}%
2004         \fi
2005         \numpagelinesR=\z@
2006         \parledgroup@correction@notespacing@init

```

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

```

2007         \clearl@drightpage}

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.

2008         \checkraw@text
2009         \ifaraw@text
2010             \getlinesfrompagelistL
2011             \getlinesfrompagelistR
2012             \l@dcalc@minoftwo{\@cs@linesonpageL}{\@cs@linesonpageR}%
2013                 {\l@dminpagelines}%
2014         \fi
2015         \repeat}

```

We have now output the text from all the chunks.

```

2016         \fi

Make sure that there are no inserts hanging around.

2017         \flush@notes
2018         \flush@notesR
2019     \endgroup

```

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

```

2020     \global\l@dpscL=\z@
2021     \global\l@dpscR=\z@
2022     \global\l@dnumpestartsL=\z@
2023     \global\l@dnumpestartsR=\z@
2024     \global\instanzaLfalse
2025     \global\instanzaRfalse
2026     \ignorespaces}
2027

```

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

```

2028 \newcommand*{\ledstrutL}{\strut}
2029 \newcommand*{\ledstrutR}{\strut}
2030

```

`\cleartoevenpage` `\cleartoevenpage`, which is defined in the memoir class, is like `\clear(double)page` except that we end up on an even page. `\cleartol@devenpage` is similar except that it first checks to see if it is already on an empty page. `\clearl@dleftpage` and `\clearl@drightpage` get us onto an odd and even page, respectively, checking that we end up on the immediately next page.

```

2031 \providecommand{\cleartoevenpage}[1][\@empty]{%
2032   \clearpage
2033   \ifodd\c@page\hbox{ }#1\clearpage\fi}
2034 \newcommand*{\cleartol@devenpage}{%
2035   \ifdim\pagetotal<\topskip% on an empty page
2036   \else
2037     \clearpage
2038   \fi
2039   \ifodd\c@page\hbox{ }\clearpage\fi}
2040 \newcommand*{\clearl@dleftpage}{%
2041   \clearpage
2042   \ifodd\c@page\else
2043     \led@err@LeftOnRightPage
2044     \hbox{ }%
2045     \cleardoublepage
2046   \fi}
2047 \newcommand*{\clearl@drightpage}{%
2048   \clearpage
2049   \ifodd\c@page
2050     \led@err@RightOnLeftPage
2051     \hbox{ }%
2052     \cleartoevenpage
2053   \fi}
2054

```

`\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 0. Similarly for `\getlinesfromparlistR`.

```

\@cs@linesinparR 2055 \newcommand*{\getlinesfromparlistL}{%
2056   \ifx\linesinpar@listL\empty
2057     \gdef\@cs@linesinparL{0}%
2058   \else
2059     \gl@p\linesinpar@listL\to\@cs@linesinparL
2060   \fi}
2061 \newcommand*{\getlinesfromparlistR}{%
2062   \ifx\linesinpar@listR\empty
2063     \gdef\@cs@linesinparR{0}%
2064   \else
2065     \gl@p\linesinpar@listR\to\@cs@linesinparR
2066   \fi}

```



2067

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

```
\@cs@linesonpageR 2068 \newcommand*{\getlinesfrompagelistL}{%
2069   \ifx\linesonpage@listL\empty
2070     \gdef\@cs@linesonpageL{1000}%
2071   \else
2072     \glp\linesonpage@listL\to\@cs@linesonpageL
2073   \fi}
2074 \newcommand*{\getlinesfrompagelistR}{%
2075   \ifx\linesonpage@listR\empty
2076     \gdef\@cs@linesonpageR{1000}%
2077   \else
2078     \glp\linesonpage@listR\to\@cs@linesonpageR
2079   \fi}
2080
```

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

```
2081 \newcommand*{\@writelinesonpageL}[1]{%
2082   \edef\next{\write\linenum@out{\string\@lopL{#1}}}%
2083   \next}
2084 \newcommand*{\@writelinesonpageR}[1]{%
2085   \edef\next{\write\linenum@outR{\string\@lopR{#1}}}%
2086   \next}
2087
```

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

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

```
2088 \newcommand*{\l@dcalc@maxoftwo}[3]{%
2089   \ifnum #2>#1\relax
2090     #3=#2\relax
2091   \else
2092     #3=#1\relax
2093   \fi}
2094 \newcommand*{\l@dcalc@minoftwo}[3]{%
2095   \ifnum #2<#1\relax
2096     #3=#2\relax
2097   \else
2098     #3=#1\relax
2099   \fi}
2100
```

`\ifl@dsamepage` `\checkpageL` tests if the space and lines already taken on the page by text and foot-  
`\l@dsamepagetrue` notes is less than the constraints. If so, then `\ifl@dpagetrue` is set FALSE and  
`\l@dsamepagefalse`

```
\ifl@dpagetrue
\l@dpagetrue
\l@dpagetruefalse
\l@dpagetruefalse
\checkpageL
\checkpageR
```

\ifl@dsamepage 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 the maximum number of lines have been output then both \ifl@dpagfull and \ifl@dsamepage are set FALSE.

```

2101 \newif\ifl@dsamepage
2102 \l@dsamepagetrue
2103 \newif\ifl@dpagfull
2104
2105 \newcommand*{\checkpageL}{%
2106   \l@dpagfulltrue
2107   \l@dsamepagetrue
2108   \check@goal
2109   \ifdim\pagetotal<\ledthegoal
2110     \ifnum\numpagelinesL<\l@dminpagelines
2111       \else
2112         \l@dsamepagefalse
2113         \l@dpagfullfalse
2114       \fi
2115     \else
2116       \l@dsamepagefalse
2117       \l@dpagfulltrue
2118     \fi}
2119 \newcommand*{\checkpageR}{%
2120   \l@dpagfulltrue
2121   \l@dsamepagetrue
2122   \check@goal
2123   \ifdim\pagetotal<\ledthegoal
2124     \ifnum\numpagelinesR<\l@dminpagelines
2125       \else
2126         \l@dsamepagefalse
2127         \l@dpagfullfalse
2128       \fi
2129     \else
2130       \l@dsamepagefalse
2131       \l@dpagfulltrue
2132     \fi}
2133

```

\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.

```

2134 \newcommand{\checkpbL}{
2135   \IfStrEq{\led@pb@setting}{after}{
2136     \xifinlistcs{\the\absline@num}{\l@prev@pb}{\l@dpagfulltrue\l@dsamepagefalse}{
2137       \xifinlistcs{\the\absline@num}{\l@prev@nopb}{\l@dpagfullfalse\l@dsamepagetrue}{
2138     }}
2139   \IfStrEq{\led@pb@setting}{before}{
2140     \numdef{\next@absline}{\the\absline@num+1}
2141     \xifinlistcs{\next@absline}{\l@prev@pb}{\l@dpagfulltrue\l@dsamepagefalse}{

```

```

2142     \xifinlistcs{\next@absline}{\l@prev@nopb}{\l@dpagfullfalse\l@dsamepagetrue}{\l@dpagfulltrue\l@dsamepagetrue}{\l@dsamepagetrue}{\l@dsamepagetrue}}
2143   }{}
2144 }
2145
2146 \newcommand{\checkpbR}{
2147   \IfStrEq{\led@pb@setting}{after}{
2148     \xifinlistcs{\the\absline@numR}{\l@prev@pbR}{\l@dpagfulltrue\l@dsamepagetrue}{\l@dsamepagetrue}{\l@dsamepagetrue}}
2149     \xifinlistcs{\the\absline@numR}{\l@prev@nopbR}{\l@dpagfullfalse\l@dsamepagetrue}{\l@dsamepagetrue}{\l@dsamepagetrue}}
2150   }{}
2151   \IfStrEq{\led@pb@setting}{before}{
2152     \numdef{\next@abslineR}{\the\absline@numR+1}
2153     \xifinlistcs{\next@abslineR}{\l@prev@pbR}{\l@dpagfulltrue\l@dsamepagetrue}{\l@dsamepagetrue}{\l@dsamepagetrue}}
2154     \xifinlistcs{\next@abslineR}{\l@prev@nopbR}{\l@dpagfullfalse\l@dsamepagetrue}{\l@dsamepagetrue}{\l@dsamepagetrue}}
2155   }{}
2156 }

```

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

```

2157 \newcommand{\checkverseL}{
2158   \ifinstanzaL
2159     \iflednopbinverse
2160       \ifinserthangingsymbol
2161         \numgdef{\prev@abslineverse}{\the\absline@num-1}
2162         \IfStrEq{\led@pb@setting}{after}{\lednopbnum{\prev@abslineverse}}{}
2163         \IfStrEq{\led@pb@setting}{before}{\ifnum\numpagelinesL<3\ledpbnum{\prev@abslineverse}\fi}{}
2164       \fi
2165     \fi
2166   \fi
2167 }
2168 \newcommand{\checkverseR}{
2169   \ifinstanzaR
2170     \iflednopbinverse
2171       \ifinserthangingsymbolR
2172         \numgdef{\prev@abslineverse}{\the\absline@numR-1}
2173         \IfStrEq{\led@pb@setting}{after}{\lednopbnumR{\prev@abslineverse}}{}
2174         \IfStrEq{\led@pb@setting}{before}{\ifnum\numpagelinesR<3\ledpbnumR{\prev@abslineverse}\fi}{}
2175       \fi
2176     \fi
2177   \fi
2178 }

```

`\ledthegoal` `\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`.  
`\check@goal` `\ledthegoal` is calculated via `\check@goal`.

```

2179 \newdimen\ledthegoal
2180 \ifshiftedpstarts
2181   \newcommand*{\goalfraction}{0.95}
2182 \else
2183   \newcommand*{\goalfraction}{0.9}

```

```

2184 \fi
2185
2186 \newcommand*{\check@goal}{%
2187   \ledthegoal=\goalfraction\pagegoal}
2188

```

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

```

\ifwrittenlinesL 2189 \newif\ifwrittenlinesL
2190 \newif\ifwrittenlinesR
2191

```

`\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.

```

2192 \newcommand*{\get@nextboxL}{%
2193   \ifvbox\namebox{\l@dLcolrawbox\the\l@dpscl}% box is not empty

   The current box is not empty; do nothing.

2194   \else%
                                     box is empty

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

2195   \ifnum\usenamecount{\l@dmaxlinesinpar\the\l@dpscl}>\@donetotallinesL
2196     \parledgroup@notes@endL
2197   \else

   Sufficient lines have been output.

2198   \ifnum\usenamecount{\l@dmaxlinesinpar\the\l@dpscl}=\@donetotallinesL
2199     \parledgroup@notes@endL
2200   \fi
2201   \ifwrittenlinesL\else

   Write out the number of lines done, and set the boolean so this is only done once.

2202   \@writelinesinparL
2203   \writtenlinesLtrue
2204   \fi
2205   \ifnum\l@dnumpstartsL>\l@dpscl

   There are still unprocessed boxes. Recalculate the maximum number of lines
   needed, and move onto the next box (by incrementing \l@dpscl). If needed, restart
   the line numbering. Increment the pstartL counter.

2206   \writtenlinesLfalse
2207   \ifbypstart@
2208     \ifnum\value{pstartL}<\value{pstartLold}
2209     \else
2210       \global\line@num=0
2211       \resetprevline@
2212     \fi
2213   \fi

2214 % Add the content of the optional argument of the previous \cs{pend}.
2215 %   \begin{macrocode}
2216   \csuse{after@pendL@\the\l@dpscl}%
2217   \global\csundef{after@pendL@\the\l@dpscl}%

```

```

2218 % \end{macrocode}
2219 % \begin{macrocode}
2220 \addtocounter{pstartL}{1}
2221 \global\pstartnumtrue
2222 \l@dcalcl@maxoftwo{\the\usernamecount{l@dmaxlinesinpar\the\l@dpscL}}%
2223 \the\@donetotallinesL}%
2224 \usernamecount{l@dmaxlinesinpar\the\l@dpscL}}%
2225 \global\@donetotallinesL \z@
2226 \global\advance\l@dpscL \@ne

Add notes of parallel ledgroup.
2227 \parledgroup@notes@endL
2228 \parledgroup@correction@notespadding@final{L}
2229 \else
2230 % Add the content of the optional argument of the last \cs{pend}.
2231 % \begin{macrocode}
2232 \l@dpagefulltrue
2233 \csuse{after@pendL@the\l@dpscL}%
2234 \global\csundef{after@pendL@the\l@dpscL}%
2235 % \end{macrocode}
2236 % \begin{macrocode}
2237 \fi
2238 \fi
2239 \fi}

2240 \newcommand*{\get@nextboxR}{%
2241 \ifvbox\namebox{l@dRcolrawbox\the\l@dpscR}% box is not empty
2242 \else% box is empty
2243 \ifnum\usernamecount{l@dmaxlinesinpar\the\l@dpscR}>\@donetotallinesR
2244 \parledgroup@notes@endR
2245 \else
2246 \ifnum\usernamecount{l@dmaxlinesinpar\the\l@dpscR}=\@donetotallinesR
2247 \parledgroup@notes@endR
2248 \fi
2249 \ifwrittenlinesR\else
2250 \@writelinesinparR
2251 \writtenlinesRtrue
2252 \fi
2253 \ifnum\l@dnumpstartsR>\l@dpscR
2254 \writtenlinesRfalse
2255 \ifbypstart@R
2256 \ifnum\value{pstartR}<\value{pstartRold}
2257 \else
2258 \global\line@numR=0
2259 \resetprevline@
2260 \fi
2261 \fi
2262 \csuse{after@pendR@the\l@dpscR}%
2263 \global\csundef{after@pendR@the\l@dpscR}%
2264 \addtocounter{pstartR}{1}
2265 \global\pstartnumRtrue

```

```

2266      \l@dcalcm@maxoftwo{\the\usernamecount{l@dmaxlinesinpar\the\l@dpscR}}}%
2267      {\the\@donetotallinesR}%
2268      {\usernamecount{l@dmaxlinesinpar\the\l@dpscR}}}%
2269      \global\@donetotallinesR \z@
2270      \global\advance\l@dpscR \@ne
2271      \parledgroup@notes@endR
2272      \parledgroup@correction@notes@spacing@final{R}
2273      \else
2274      \csuse{after@pendR@the\l@dpscR}%
2275      \global\csundef{after@pendR@the\l@dpscR}%
2276      \l@dpagfulltrue
2277      \fi
2278      \fi
2279      \fi}
2280

```

## 28 Sections' titles' commands

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

```

2281 \newcommand{\eledsectnotoc}[1]{\xdef\@eledsectnotoc{#1}}
2282 \eledsectnotoc{R}

```

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

```

2283 \newcommand{\eledsectmark}[1]{\xdef\@eledsectmark{#1}}
2284 \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`.

```

2285 \newskip\eledsection@correcting@skip

```

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

```

2286 \newwrite\eled@sectioningR@out

```

## 29 Page break/no page break, depending on the specific line

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

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

```

2287 \def\l@prev@pbR{}
2288 \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.

```
2289 \newcommand{\ledpbR}{\write\linenum@outR{\string\led@pbR}}
2290 \newcommand{\ledpbnR}[1]{\write\linenum@outR{\string\led@pbnR{#1}}}
2291 \newcommand{\lednopbR}{\write\linenum@outR{\string\led@nopbR}}
2292 \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@pbnR` add the argument in the `\prev@pbR` list. The `\led@nopbR` add the absolute line number in the `\prev@nopbR` list. The `\led@nopbnR` add the argument in the `\prev@nopbR` list.

```
2293 \newcommand{\led@pbR}{\listcsxadd{1@prev@pbR}{\the\absline@numR}}
2294 \newcommand{\led@pbnR}[1]{\listcsxadd{1@prev@pbR}{#1}}
2295 \newcommand{\led@nopbR}{\listcsxadd{1@prev@nopbR}{\the\absline@numR}}
2296 \newcommand{\led@nopbnR}[1]{\listcsxadd{1@prev@nopbR}{#1}}
```

### 30 Parallel ledgroup

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

```
2297 \newmarks\parledgroup@
2298 \newmarks\parledgroup@series
2299 \newmarks\parledgroup@type
```

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

```
2300 \newcommand{\parledgroup@notes@startL}{%
2301   \ifnum\usenamecount{1@dmaxlinesinpar\the\l@dpscL}>0%
2302     \IfStrEq{\splitfirstmarks\parledgroup@type}{footnoteX}{\csuse{hooknoteX@\splitfirstmarks\parledgroup@type}{footnoteX}}{\csuse{hooknoteX@\splitfirstmarks\parledgroup@type}{footnoteX}}
2303     \IfStrEq{\splitfirstmarks\parledgroup@type}{Xfootnote}{\csuse{hookXnote@\splitfirstmarks\parledgroup@type}{Xfootnote}}{\csuse{hookXnote@\splitfirstmarks\parledgroup@type}{Xfootnote}}
2304   \fi%
2305   \global\ledgroupnotesL@true%
2306   \insert@noterule@ledgroup{L}%
2307 }
2308 \newcommand{\parledgroup@notes@startR}{%
2309   \ifnum\usenamecount{1@dmaxlinesinpar\the\l@dpscR}>0%
2310     \IfStrEq{\splitfirstmarks\parledgroup@type}{footnoteX}{\csuse{hooknoteX@\splitfirstmarks\parledgroup@type}{footnoteX}}{\csuse{hooknoteX@\splitfirstmarks\parledgroup@type}{footnoteX}}
2311     \IfStrEq{\splitfirstmarks\parledgroup@type}{Xfootnote}{\csuse{hookXnote@\splitfirstmarks\parledgroup@type}{Xfootnote}}{\csuse{hookXnote@\splitfirstmarks\parledgroup@type}{Xfootnote}}
2312   \fi%
2313   \global\ledgroupnotesR@true%
2314   \insert@noterule@ledgroup{R}%
2315 }
```

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

```
2316 \newcommand{\parledgroup@notes@endL}{%
2317   \global\ledgroupnotesL@false%
2318 }
2319 \newcommand{\parledgroup@notes@endR}{%
2320   \global\ledgroupnotesR@false%
2321 }
```

`\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`

```
2322 \newcommand{\insert@noterule@ledgroup}[1]{
2323   \IfStrEq{\splitbotmarks\parledgroup@}{begin}{%
2324     \IfStrEq{\splitbotmarks\parledgroup@type}{Xfootnote}{
2325       \csuse{ifledgroupnotes#1@}
2326       \vskip\skip\csuse{mp\splitbotmarks\parledgroup@series footins}
2327       \csuse{\splitbotmarks\parledgroup@series footnoterule}
2328       \fi
2329     }
2330   {}
2331   \IfStrEq{\splitbotmarks\parledgroup@type}{footnoteX}{
2332     \csuse{ifledgroupnotes#1@}
2333     \vskip\skip\csuse{mpfootins\splitbotmarks\parledgroup@series}
2334     \csuse{footnoterule\splitbotmarks\parledgroup@series}
2335     \fi
2336   }{}
2337 }
2338 {}
2339 }
```

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

```
2340 \newcommand{\parledgroupnotespadding}{}%
```

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

```
2341 \dimdef{\parledgroup@notespadding@correction}{0pt}
2342 \newcommand{\parledgroup@notespadding@set@correction}{%
2343   {\notefontsetup\parledgroupnotespadding\dimgdef{\temp@spacing}{\baselineskip}}%
2344   \dimgdef{\parledgroup@notespadding@correction}{\baselineskip-\temp@spacing}%
2345 }
```

`\parledgroup@correction@notespadding@init` `\parledgroup@correction@notespadding@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.

```
2346 \newcommand{\parledgroup@correction@notespadding@init}{%
```



```

2347 \dimdef{\parledgroup@notespacing@correction@accumulated}{0pt}
2348 \dimdef{\parledgroup@notespacing@correction@modulo}{0pt}
2349 }
2350 \parledgroup@correction@notespacing@init

```

`\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`.

```

2351 \newcommand{\parledgroup@correction@notespacing@final}[1]{
2352   \ifparledgroup
2353   \vspace{\parledgroup@notespacing@correction@accumulated}
2354   \parledgroup@correction@notespacing@init%
2355   \ifstrequal{#1}{L}{
2356     \numdef{\@checking}{\the\l@dpscL-1}
2357   }{
2358     \numdef{\@checking}{\the\l@dpscR-1}
2359   }
2360   \dimdef{\@beforenotes@current@diff}{\csuse{@parledgroup@beforenotes@\@checking L}-\csuse{@parledgroup@beforenotes@\@checking R}}
2361   \ifstrequal{#1}{L}{
2362     {% Left
2363       \ifdimgreater{\@beforenotes@current@diff}{0pt}{\vspace{-\@beforenotes@current@diff}}%
2364     }%
2365     {% Right
2366       \ifdimgreater{\@beforenotes@current@diff}{0pt}{\vspace{\@beforenotes@current@diff}}%
2367     }%
2368   \fi
2369 }

```

`\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.

```

2370 {}
2371 \newcommand{\parledgroup@correction@notespacing}[1]{%
2372   \csuse{ifledgroupnotes#1@}%
2373   \vspace{-\parledgroup@notespacing@correction}%
2374   \dimdef{\parledgroup@notespacing@correction@accumulated}{\parledgroup@notespacing@correction@accumulated}
2375   \dimdef{\parledgroup@notespacing@correction@modulo}{\parledgroup@notespacing@correction@modulo}

```

```

2376     \ifdimless{\parledgroup@notespacing@correction@modulo}{\baselineskip}{\advance\num
2377     \dimdef{\parledgroup@notespacing@correction@modulo}{\parledgroup@notespacing@correct
2378     }% mean greater than equal
2379     \fi%
2380 }

```

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

```

2381 \dimdef\parledgroup@beforenotesL{0pt}
2382 \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.

```

2383 \newcommand{\parledgroup@beforenotes@save}[1]{
2384   \ifparledgroup
2385     \csdimgdef{@parledgroup@beforenotes@the\csuse{1@dnumstarts#1}#1}{\csuse{parledgroup
2386     \csdimgdef{parledgroup@beforenotes#1}{0pt}
2387   \fi
2388 }

```

## 31 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 `\setstanzaindent` 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.

<b>Symbols</b>	<code>\@adv</code> . . . . .	352, 633, 634
<code>\&amp;</code> . . . . .	1564, 1565, 1569, 1586, 1607	<code>\@afterindentfalse</code> . . . . . 758
<code>\@M</code> . . . . .	1575	<code>\@arabic</code> . . . . . 208, 209, 807, 810



- \affixline@num ..... 974  
 \affixline@numR ..... 1039, 1167  
 \affixpstart@numL ..... 990, 1274  
 \affixpstart@numR ..... 1274  
 \affixside@note ..... 976  
 \affixside@noteR ..... 1041, 1446  
 \aftercolumnseparator . 6, 1812, 1853  
 \appto ..... 1452, 1453  
 \araw@textfalse ..... 1870  
 \araw@texttrue ..... 1870  
 astanza (environment) ..... 10, 1567  
 \AtBeginDocument ... 1423, 1657, 1683
- B**
- \ballast@count ..... 1095, 1100  
 \bbl@main@language ..... 1699, 1700  
 \bbl@set@language ..... 1690, 1691  
 \beforecolumnseparator .....  
 ..... 6, 1806, 1809, 1815, 1853  
 \beginnumbering ..... 8, 782, 821  
 \beginnumberingR ... 37, 113, 782, 865  
 \bfseries ..... 807, 810  
 \bypage@Rfalse ..... 133, 148, 153  
 \bypage@Rtrue ..... 133, 143  
 \bypstart@Rfalse ..... 133, 144, 154  
 \bypstart@Rtrue ..... 133, 149
- C**
- \c@ballast ..... 1100  
 \c@chapter ..... 95  
 \c@chapterR ..... 95  
 \c@firstlinenumR ..... 173, 1240  
 \c@firstsublinenumR ..... 177, 1235  
 \c@linenumincrementR ..... 173, 1240  
 \c@page 600, 602, 2033, 2039, 2042, 2049  
 \c@pstart ..... 1412  
 \c@pstartL ..... 807  
 \c@pstartR ..... 810, 1400  
 \c@section ..... 96  
 \c@sectionR ..... 96  
 \c@sublinenumincrementR .. 177, 1235  
 \c@subsection ..... 97  
 \c@subsectionR ..... 97  
 \c@subsubsection ..... 98  
 \c@subsubsectionR ..... 98  
 \ch@ck@l@ckR ..... 1167  
 \ch@cksub@l@ckR ..... 1167  
 \ch@cksub@lockR ..... 1236  
 \chapter ..... 744, 745, 754  
 \chapterinpages ..... 736, 745, 756
- \chardef ..... 1564  
 \check@goal ..... 2108, 2122, 2179  
 \check@pstarts .....  
 ..... 1733, 1782, 1858, 1919, 1927, 1935  
 \checkpageL ..... 1946, 1965, 2101  
 \checkpageR ..... 1977, 1996, 2101  
 \checkpb@columns ... 1778, 1820, 1824  
 \checkpbL ..... 1967, 2134  
 \checkpbR ..... 1998, 2134  
 \checkraw@text .....  
 ..... 1739, 1775, 1870, 1943, 2008  
 \checkverseL ..... 1776, 1966, 2157  
 \checkverseR ..... 1777, 1997, 2157  
 \cleardoublepage ..... 2045  
 \clearl@dleftpage ..... 1976, 2031  
 \clearl@drighpage ..... 2007, 2031  
 \cleartoevenpage ..... 2031  
 \cleartol@devenpage ..... 1908, 2031  
 \closeout ..... 87, 581, 587, 590, 594  
 \columnrulewidth ..... 5, 1844  
 \Columns ..... 5, 1718  
 \columns@position .. 1763, 1773, 1849  
 \columnseparator ..... 5, 1811, 1844  
 \columnspan ..... 6, 1849  
 \correcthangingL ..... 993, 1545  
 \correcthangingR ..... 1545  
 \countLline ..... 945, 956  
 \countRline ..... 945, 1021  
 \critext ..... 668  
 \cs ..... 1821, 2214, 2230  
 \csdingdef ..... 2385, 2386  
 \csexpandonce .. 1495, 1499, 1515, 1526  
 \csgdef ..... 854, 897, 918, 939  
 \csundef ..... 1004, 1748, 1759,  
 ..... 1952, 1983, 2217, 2234, 2263, 2275  
 \csuse ..... 1002,  
 ..... 1514, 1525, 1746, 1757, 1951,  
 ..... 1982, 2216, 2233, 2262, 2274,  
 ..... 2302, 2303, 2310, 2311, 2325–  
 ..... 2327, 2332–2334, 2360, 2372, 2385
- D**
- \DeclareOption ..... 8–10  
 \def@tempb ..... 151  
 \dimdef ..... 2341, 2347, 2348,  
 ..... 2360, 2374, 2375, 2377, 2381, 2382  
 \dimen ..... 619, 620, 624–626, 630  
 \dingdef ..... 2343, 2344  
 \divide ..... 1172  
 \do@actions ..... 1077

- `\do@actions@fixedcodeR` ..... 1104  
`\do@actions@nextR` ..... 1104  
`\do@actionsR` ..... 1059, 1104  
`\do@ballast` ..... 1078  
`\do@ballastR` ..... 1060, 1095  
`\do@insidelineLhook` ..... 993, 1014  
`\do@insidelineRhook` ..... 1014  
`\do@lineL` ..... 955, 1742, 1953  
`\do@lineLhook` ..... 960, 1014  
`\do@lineR` ..... 1019, 1753, 1984  
`\do@lineRhook` ..... 1014, 1025  
`\do@lockoff` ..... 494  
`\do@lockoffL` ..... 518  
`\do@lockoffR` ..... 494  
`\do@lockon` ..... 459  
`\do@lockonL` ..... 491  
`\do@lockonR` ..... 459  
`\dolistloop` ..... 1457, 1476, 1483  
`\dummy@ref` ..... 540
- E**
- `\edfont@info` ..... 719, 722, 728, 731  
`\edlabel` ..... 1388  
`\edtext` ..... 691  
`\eled@sectioningR@out` .. 64, 87, 2286  
`\eled@sections@@` 977, 1000, 1743, 1954  
`\eled@sectionsR@@` 62, 1042, 1754, 1985  
`\eledmac@error` ..... 22, 25, 28, 30  
`\eledmac@warning` ... 1458, 1477, 1484  
`\eledsection@correcting@skip` ...  
     ..... 1012, 1719, 1899, 2285  
`\eledsectmark` ..... 14, 2283  
`\eledsectnotoc` ..... 14, 2281  
`\empty` .. 73, 76, 258, 266, 681, 704,  
     717, 726, 830, 874, 1157, 1239,  
     1247, 1343–1345, 1356, 1367,  
     1391, 1403, 2056, 2062, 2069, 2075  
`\end@lemmas` ..... 681, 682, 704, 705  
`\endashchar` ..... 1378  
`\endgraf` ..... 907, 928, 1727, 1911  
`\endline@num` ..... 547, 553  
`\endlock` ..... 652, 1573, 1582, 1587  
`\endnumbering` ..... 8, 66, 117, 783  
`\endnumberingR` 40, 66, 101, 112, 125, 783  
`\endpage@num` ..... 546, 553  
`\endstanzaextra` ..... 1589  
`\endsub` ..... 619  
`\endsubline@num` ..... 548, 554  
`\enlargethispage` ..... 1841
- environments:
- `astanza` ..... 10, 1567  
`Leftside` ..... 7, 763  
`pages` ..... 6, 736  
`pairs` ..... 5, 736  
`Rightside` ..... 7, 780  
`\extensionchars` .... 57, 107, 122, 130
- F**
- `\fx@l@cksR` ..... 1167  
`\first@linenum@out@Rfalse` .. 576, 582  
`\first@linenum@out@Rtrue` ..... 576  
`\firstlinenum` ..... 7, 182  
`\firstlinenum*` ..... 7, 182  
`\firstsublinenum` ..... 7, 182  
`\firstsublinenum*` ..... 7, 182  
`\fix@page` ..... 319, 326  
`\flag@end` .....  
     . 603, 677, 687, 688, 700, 710, 711  
`\flag@start` .....  
     . 603, 676, 677, 688, 699, 700, 711  
`\flush@notes` ..... 1794, 2017  
`\flush@notesR` ..... 1365, 1795, 2018  
`\footnotelang@lua` ..... 1509, 1520  
`\footnotelang@poly` ..... 1513, 1524  
`\fullstop` . 221, 1375, 1377, 1379, 1381
- G**
- `\get@linelistfile` ..... 254  
`\get@nextboxL` ..... 1964, 2192  
`\get@nextboxR` ..... 1995, 2192  
`\getline@numL` ..... 966, 1075  
`\getline@numR` ..... 1031, 1057  
`\getlinesfrompagelistL` .....  
     ..... 1931, 2010, 2068  
`\getlinesfrompagelistR` .....  
     ..... 1932, 2011, 2068  
`\getlinesfromparlistL` ... 1923, 2055  
`\getlinesfromparlistR` ... 1924, 2055  
`\gl@p` ..... 261, 262,  
     269, 270, 682, 705, 721, 730,  
     1160, 1161, 1349, 1353, 1368,  
     1394, 1406, 2059, 2065, 2072, 2078  
`\goalfraction` ..... 7, 2179
- H**
- `\hangingsymbol` ..... 11, 1536, 1542  
`\hb@xt@` ..... 973, 984, 993, 1038,  
     1050, 1762, 1959, 1962, 1990, 1993

\hsize ..... 848,  
891, 1762, 1959, 1962, 1990, 1993

## I

\if@filesw ..... 1671  
\if@firstcolumn 1253, 1278, 1299, 1462  
\if@ledgroup ..... 586  
\if@nobreak ..... 813, 857  
\if@nopb ..... 1826, 1841  
\if@pb ..... 1825, 1842  
\if@pstarts .... 1734, 1858, 1920, 1936  
\if@RTL ..... 677, 688, 700, 711, 1005  
\ifaraw@text ... 1740, 1870, 1945, 2009  
\ifautopar ..... 840, 884  
\ifbypage@ ..... 342  
\ifbypage@R ..... 133, 332, 1139  
\ifbypstart@ ..... 561, 1783, 2207  
\ifbypstart@R ... 133, 565, 1787, 2255  
\ifdefstring ..... 997, 998,  
1745, 1756, 1763, 1773, 1950, 1981  
\ifdim ... 620, 624, 626, 630, 1806,  
1812, 1959, 1990, 2035, 2109, 2123  
\ifdimgreater ..... 2363, 2366  
\ifdimless ..... 2376  
\iffirst@linenum@out@R .... 576, 580  
\ifhbox ..... 1765, 1770  
\ifinserthangingsymbol .....  
..... 1534, 1548, 2160  
\ifinserthangingsymbolR .....  
..... 1532, 1540, 1558, 2171  
\ifinstanzaL 761, 761, 1535, 1547, 2158  
\ifinstanzaR 761, 762, 1541, 1557, 2169  
\ifl@ddash ..... 1378  
\ifl@delin ..... 1380, 1381  
\ifl@desl ..... 1381  
\ifl@d@pnun ..... 1375, 1379  
\ifl@d@ssub ..... 1377  
\ifl@dpagefull ..... 1969, 2000, 2101  
\ifl@d@p@ng ..... 12, 1546, 1556  
\ifl@d@p@ring ..... 12, 70  
\ifl@dsamelang ..... 1668  
\ifl@dsamepage ..... 1949, 1979, 2101  
\ifl@dskipnumber ..... 1230  
\ifl@dusedbabel ..... 1666  
\iflabelpstart ..... 850, 893  
\ifledgroupnotesL@ ..... 1079  
\ifledgroupnotesR@ ..... 1061, 1229  
\iflednopbinverse ..... 2159, 2170  
\ifledplinenum ..... 1376

\ifledRcol ..... 12, 165, 187, 191,  
195, 199, 241, 256, 320, 329,  
354, 368, 385, 402, 414, 422,  
443, 488, 515, 524, 533, 604,  
614, 621, 627, 633, 640, 648,  
653, 657, 662, 672, 695, 716,  
1389, 1428, 1494, 1507, 1694, 1706  
\ifluatex ..... 1508, 1519  
\ifnoteschanged@ ..... 80  
\ifnumberedpar@ .....  
... 823, 867, 903, 924, 1492, 1506  
\ifnumbering ..... 138, 819, 900  
\ifnumberingR ... 38, 67, 103, 863, 921  
\ifnumberline ..... 1061, 1079, 1229  
\ifnumberpstart ... 841, 885, 912, 933  
\ifnumequal ..... 1450  
\ifnumgreater ..... 1458, 1477, 1484  
\ifodd ..... 1263, 1288,  
1309, 1472, 2033, 2039, 2042, 2049  
\ifparledgroup ..... 2352, 2384  
\ifpst@rtedL ..... 33, 827  
\ifpst@rtedR ..... 33, 871  
\ifpstartnum ..... 1319, 1324  
\ifpstartnumR ..... 1274  
\ifshiftedpstarts 5, 1958, 1989, 2180  
\ifsidepstartnum 842, 886, 1276, 1297  
\ifstreempty ..... 852, 895, 916, 937  
\IfStrEq ..... 964, 1029, 1827,  
1835, 2135, 2139, 2147, 2151,  
2162, 2163, 2173, 2174, 2302,  
2303, 2310, 2311, 2323, 2324, 2331  
\ifstrequal ..... 2355, 2361  
\ifsublines@ ..... 219,  
308, 353, 386, 393, 424, 433,  
445, 452, 464, 498, 552, 554,  
1062, 1080, 1146, 1233, 1430, 1434  
\ifvbox 957, 1022, 1874, 1877, 2193, 2241  
\ifvmode ..... 1396, 1408  
\ifwrittenlinesL ..... 2189, 2201  
\ifwrittenlinesR ..... 2190, 2249  
\initnumbering@sectcmd 109, 739, 748  
\initnumbering@sectcountR 61, 90, 1980  
\InputIfFileExists ..... 63  
\insert@count .....  
... 530, 610, 673, 696, 1501, 1528  
\insert@countR .....  
... 531, 606, 672, 695, 1497, 1517  
\insert@noterule@ledgroup .....  
..... 2306, 2314, 2322  
\inserthangingsymbolfalse ..... 970

- \inserthangingsymbolL .... 993, 1532
  - \inserthangingsymbolR ..... 1532
  - \inserthangingsymbolRfalse .... 1035
  - \inserthangingsymbolRtrue .... 1033
  - \inserthangingsymboltrue ..... 968
  - \insertlines@listR .....
    - .... 73, 229, 243, 536, 1345, 1349
  - \inserts@list ..... 829, 1500, 1527
  - \inserts@listR ..... 873, 1340,
    - 1343, 1353, 1367, 1368, 1496, 1516
  - \istanzaLfalse ..... 1802, 2024
  - \istanzaLtrue ..... 772
  - \istanzaRfalse ..... 1803, 2025
  - \istanzaRtrue ..... 795
  - \interlinepenalty ..... 1575
  - \itemcount@ ..... 1448, 1450,
    - 1455, 1458, 1475, 1477, 1482, 1484
- L**
- \l@d@nums ..... 719, 722, 728, 731
  - \l@d@set ..... 401, 648, 649
  - \l@dbbl@set@language .... 1668, 1691
  - \l@dbfnote ..... 1491
  - \l@dc@maxchunks ..... 835, 837,
    - 879, 881, 1625, 1635, 1643, 1650
  - \l@dcalc@maxoftwo .....
    - .... 1925, 2088, 2222, 2266
  - \l@dcalc@minoftwo .. 1933, 2012, 2088
  - \l@dcalcnun ..... 1167
  - \l@dchecklang ..... 1668
  - \l@dchset@num ..... 275, 278, 401
  - \l@dcsnotetext ..... 1457, 1460
  - \l@dcsnotetext@l ..... 1476
  - \l@dcsnotetext@r ..... 1483
  - \l@emptyd@ta ..... 961, 1026
  - \l@dend@stuff ..... 58, 108, 123, 131
  - \l@dgetline@margin ..... 163
  - \l@dgetsidenote@margin ..... 1439
  - \l@dld@ta ..... 991,
    - 1254, 1266, 1279, 1291, 1300, 1312
  - \l@dleftbox .....
    - .. 942, 972, 984, 1764, 1959, 1962
  - \l@dlinenumR ..... 211
  - \l@dlsn@te ..... 992
  - \l@dmake@labels ..... 1412
  - \l@dmake@labelsR ..... 1400, 1417
  - \l@dminpagelines .....
    - .... 1894, 1934, 2013, 2110, 2124
  - \l@dnumpstartsL .....
    - .... 834, 835, 837, 839, 854,
  - 918, 1629, 1661, 1722, 1723,
    - 1799, 1861, 1905, 1906, 2022, 2205
  - \l@dnumpstartsR .....
    - .. 42, 878, 879, 881, 883, 897,
    - 939, 1629, 1662, 1722, 1723,
    - 1800, 1864, 1905, 1906, 2023, 2253
  - \l@doldbbl@set@language ..... 1690
  - \l@doldselectlanguage 1689, 1693, 1698
  - \l@dpagfullfalse .....
    - .... 2101, 2137, 2142, 2149, 2154
  - \l@dpagfulltrue ..... 2101,
    - 2136, 2141, 2148, 2153, 2232, 2276
  - \l@dpaddingfalse ..... 14, 738, 753
  - \l@dpaddingtrue ..... 747
  - \l@dpairingfalse ..... 12, 741, 752
  - \l@dpairingtrue ..... 737, 746
  - \l@dpscL ..... 957,
    - 962, 977, 999, 1002, 1004, 1631,
    - 1663, 1731, 1737, 1743, 1746,
    - 1748, 1797, 1861, 1874, 1915,
    - 1921, 1926, 1929, 1937, 1951,
    - 1952, 1954, 2020, 2193, 2195,
    - 2198, 2205, 2216, 2217, 2222,
    - 2224, 2226, 2233, 2234, 2301, 2356
  - \l@dpscR ..... 1022, 1027, 1042,
    - 1632, 1664, 1732, 1738, 1754,
    - 1757, 1759, 1798, 1864, 1877,
    - 1916, 1922, 1930, 1938, 1982,
    - 1983, 1985, 2021, 2241, 2243,
    - 2246, 2253, 2262, 2263, 2266,
    - 2268, 2270, 2274, 2275, 2309, 2358
  - \l@drd@ta ..... 993,
    - 1256, 1264, 1281, 1289, 1302, 1310
  - \l@drightbox .....
    - 942, 1037, 1050, 1769, 1990, 1993
  - \l@drsn@te ..... 994
  - \l@dsamepagefalse .....
    - .... 2101, 2136, 2141, 2148, 2153
  - \l@dsamepagetrue .....
    - .... 2101, 2137, 2142, 2149, 2154
  - \l@dsetupmaxlinecounts .. 1642, 1659
  - \l@dsetuprawboxes ..... 1634, 1658
  - \l@dskipnumberfalse ..... 1231
  - \l@dskipnumbertrue ..... 1127
  - \l@dunhbox@line ..... 993, 1007, 1010
  - \l@dusedbabelfalse ..... 1666, 1686
  - \l@dusedbabeltrue ..... 1666, 1688
  - \l@duselanguage .....
    - .... 1678, 1741, 1752, 1947, 1978
  - \l@dzeromaxlinecounts ... 1642, 1660



- \l@dzeroopenalties 906, 927, 1726, 1910
- \l@prev@nopbR ..... 49, 2288
- \l@prev@pbR ..... 48, 2287
- \l@pscl ..... 1631
- \l@pscR ..... 1631
- \label@refs .....
  - 1392, 1394, 1400, 1404, 1406, 1412
- \labelref@list ..... 1403, 1406, 1435
- \labelref@listR 1386, 1391, 1394, 1431
- \language name .. 1669, 1670, 1672–1675
- \last@page@num ..... 340, 346
- \last@page@numR ..... 326
- \lastbox ..... 965, 1030
- \lastskip ..... 619, 625
- \lcolwidth .....
  - 5, 7, 16, 749, 848, 973, 984, 1749
- \led@err@BadLeftRightPstarts ...
  - ..... 24, 1723, 1906
- \led@err@LeftOnRightPage ... 27, 2043
- \led@err@LineationInNumbered ... 139
- \led@err@NumberingNotStarted ... 84
- \led@err@numberingShouldHaveStarted
  - ..... 111
- \led@err@NumberingStarted ..... 39
- \led@err@PendNoPstart ..... 904, 925
- \led@err@PendNotNumbered ... 901, 922
- \led@err@PstartInPstart ... 824, 868
- \led@err@PstartNotNumbered . 820, 864
- \led@err@RightOnLeftPage ... 27, 2050
- \led@err@TooManyPstarts 21, 836, 880
- \led@mess@NotesChanged ..... 81
- \led@mess@SectionContinued .....
  - ..... 106, 121, 129
- \led@nopbnumR ..... 2292, 2293
- \led@nopbR ..... 2291, 2293
- \led@pb@setting .....
  - ..... 1827, 1835, 2135, 2139,
  - 2147, 2151, 2162, 2163, 2173, 2174
- \led@pbnumR ..... 2290, 2293
- \led@pbR ..... 2289, 2293
- \led@warn@BadAction ..... 1129
- \led@warn@BadAdvancelineLine 371, 377
- \led@warn@BadAdvancelineSubline .
  - ..... 357, 363
- \led@warn@BadLineation ..... 156
- \led@warn@BadSetline ..... 638
- \led@warn@BadSetlinenum ..... 646
- \led@warn@DuplicateLabel ..... 1419
- \ledgroupnotesL@false ..... 2317
- \ledgroupnotesL@true ..... 2305
- \ledgroupnotesR@false ..... 2320
- \ledgroupnotesR@true ..... 2313
- \ledllfill ..... 993
- \lednopb ..... 791
- \lednopbnum ..... 2162, 2289
- \lednopbnumR ..... 2173, 2289
- \lednopbR ..... 791, 2291
- \ledpb ..... 790
- \ledpbnum ..... 2163
- \ledpbnumR ..... 2174, 2289
- \ledpbR ..... 790, 2289
- \ledRcol@false ..... 1053
- \ledRcol@true ..... 1020
- \ledRcolfalse ..... 15, 764, 797
- \ledRcoltrue ..... 781
- \ledrlfill ..... 993
- \ledsavedprintlines ..... 9, 1373
- \ledsectnomark ..... 998
- \ledsectnotoc ..... 997, 1950, 1981
- \ledstrutL ..... 1959, 1962, 2028
- \ledstrutR ..... 1990, 1993, 2028
- \ledthegoal ..... 2109, 2123, 2179
- \leftlinenumR ..... 211, 1254, 1266
- \leftpstartnumL ..... 1274
- \leftpstartnumR ..... 1274
- Leftside (environment) ..... 7, 763
- \Leftsidehook ..... 770, 775
- \Leftsidehookend ..... 774, 775
- \line@list ..... 726, 730
- \line@list@stuff ..... 122
- \line@list@stuffR .. 57, 107, 130, 578
- \line@listR . 76, 229, 242, 554, 717, 721
- \line@margin ..... 168, 1284
- \line@marginR ..... 161, 1259, 1305
- \line@num . 343, 375, 376, 378, 396,
  - 407, 408, 436, 561, 1086, 1433, 2210
- \line@numR . 50, 218, 225, 280, 314,
  - 333, 369, 370, 372, 389, 403,
  - 404, 427, 547, 551, 565, 1068,
  - 1140, 1149, 1238, 1240, 1242,
  - 1243, 1429, 1458, 1477, 1484, 2258
- \lineation ..... 792
- \lineationR ..... 137, 792
- \linenum@out ..... 600, 609,
  - 617, 622, 628, 634, 641, 649,
  - 654, 658, 1402, 1784, 1885, 2082
- \linenum@outR ..... 575, 581,
  - 583, 587, 588, 590, 591, 594,
  - 595, 602, 605, 615, 621, 627,

- 633, 640, 648, 653, 657, 662,  
1390, 1788, 1890, 2085, 2289–2292  
`\linenumberlist` ..... 1239, 1243  
`\linenumincrement` ..... 7, 182  
`\linenumincrement*` ..... 7, 182  
`\linenummargin` ..... 161  
`\linenumr@p` .... 1376, 1380, 1429, 1433  
`\linenumrepR` ..... 208, 218  
`\linenumsep` .....  
    . 213, 215, 1321, 1324, 1333, 1336  
`\linesinpar@listL` .....  
    ..... 234, 250, 562, 2056, 2059  
`\linesinpar@listR` .....  
    ..... 234, 246, 566, 2062, 2065  
`\linesonpage@listL` 251, 570, 2069, 2072  
`\linesonpage@listR` 247, 573, 2075, 2078  
`\list@clear` .....  
    . 242–247, 250, 251, 253, 829, 873  
`\list@clearing@reg` ..... 249  
`\list@create` .....  
    ... 229–232, 234–236, 1340, 1386  
`\listcsxadd` ..... 348, 2293–2296  
`\lock@disp` ..... 1199, 1203, 1208  
`\lock@off` .... 485, 486, 494, 657, 658  
`\lock@on` ..... 653, 654
- M**
- `\managestanza@modulo` ..... 1601  
`\maxchunks` ..... 4, 1625  
`\maxlinesinpar@list` ..... 234, 253  
`\memorydump` ..... 9, 769, 786  
`\memorydumpL` ..... 116, 769  
`\memorydumpR` ..... 116, 786  
`\message` ..... 56  
`\multiply` ..... 1173
- N**
- `\n@num` ..... 522, 662  
`\n@num@reg` ..... 528  
`\namebox` ..... 957, 962, 1022,  
    1027, 1609, 1874, 1877, 2193, 2241  
`\NeedsTeXFormat` ..... 2  
`\new@line` ..... 1007, 1010  
`\new@lineL` ..... 599, 993  
`\new@lineR` ..... 601  
`\newbox` ..... 802, 942, 943, 1610  
`\newcommandx` ..... 812, 856, 899, 920  
`\newcounter` ..... 90–93, 173,  
    175, 177, 179, 805, 806, 808, 809  
`\newif` ..... 5,  
    13, 34, 133, 134, 576, 761, 762,  
    1328, 1532, 1666, 1825, 1826,  
    1858, 1870, 2101, 2103, 2189, 2190  
`\newlength` ..... 1853, 1856  
`\newmarks` ..... 2297–2299  
`\newnamebox` ..... 1609, 1637, 1638  
`\newnamecount` ..... 1620, 1645  
`\newsavebox` ..... 1716, 1717  
`\newwrite` ..... 575, 2286  
`\next@absline` .....  
    .... 1828, 1830, 1832, 2140–2142  
`\next@abslineR` .....  
    .... 1829, 1831, 1833, 2152–2154  
`\next@action` ..... 270  
`\next@actionline` ..... 267, 269  
`\next@actionlineR` .....  
    . 259, 261, 1098, 1136, 1158, 1160  
`\next@actionR` ..... 262, 1099,  
    1137, 1138, 1143, 1144, 1152, 1161  
`\next@insert` ..... 830  
`\next@insertR` .....  
    874, 1344, 1347, 1349, 1352, 1356  
`\next@page@num` ..... 347, 419  
`\next@page@numR` 54, 283, 285, 337, 416  
`\no@expands` ..... 670, 693  
`\noindent` ..... 854, 897, 918, 939  
`\normal@page@breakR` ..... 47  
`\normal@pars` ..... 69, 833, 877  
`\normalbfnoteX` ..... 1505  
`\notefontsetup` ..... 2343  
`\noteschanged@true` .....  
    ..... 74, 77, 718, 727, 1346  
`\num@lines` ..... 907, 1727, 1911  
`\num@linesR` ..... 801, 928, 1728, 1912  
`\numberedpar@true` ..... 849, 892  
`\numberingRfalse` ..... 68  
`\numberingRtrue` ..... 44, 101, 126  
`\numberingtrue` ..... 118  
`\numberpstartfalse` ..... 10  
`\numberpstarttrue` ..... 10  
`\numdef` 999, 1448, 1455, 1475, 1482,  
    1828, 1829, 2140, 2152, 2356, 2358  
`\numgdef` ..... 2161, 2172  
`\numlabfont` ..... 218  
`\numpagelinesL` ..... 1894,  
    1957, 1970, 1974, 2110, 2163, 2376  
`\numpagelinesR` .....  
    1894, 1988, 2001, 2005, 2124, 2174

## O

`\old@otherlanguage` ..... 1703  
`\old@startstanza` .. 771, 772, 794, 795  
`\oldchapter` ..... 744, 754  
`\one@line` ... 962, 965, 993, 1007, 1010  
`\one@lineR` ..... 801, 1027, 1030  
`\openout` ..... 64, 583, 588, 591, 595  
`\otherlanguage` ..... 1703, 1704

## P

`\p@pstartL` ..... 851  
`\p@pstartR` ..... 894  
`\page@action` ..... 284, 413, 541  
`\page@num` ..... 265, 345, 1286  
`\page@numR` .....  
     . 238, 257, 335, 546, 551, 1138,  
     1261, 1307, 1458, 1470, 1477, 1484  
`\pagebreak` ..... 1842  
`\pagegoal` ..... 2187  
`\Pages` ..... 6, 1898  
`pages` (environment) ..... 6, 736  
`\pagetotal` ..... 2035, 2109, 2123  
`pairs` (environment) ..... 5, 736  
`\paperwidth` ..... 1006, 1009  
`\par@line` ..... 908, 1729, 1913  
`\par@lineR` ..... 801, 929, 1730, 1914  
`\parbox` ..... 1749, 1760  
`\parledgroup@` .. 964, 1029, 2297, 2323  
`\parledgroup@beforenotes@save` ..  
     ..... 915, 936, 2383  
`\parledgroup@beforenotesL` ..... 2381  
`\parledgroup@beforenotesR` ..... 2381  
`\parledgroup@correction@notespacing`  
     ..... 1961, 1992, 2370  
`\parledgroup@correction@notespacing@final`  
     ..... 2228, 2272, 2351  
`\parledgroup@correction@notespacing@init`  
     ..... 1975, 2006, 2346, 2354  
`\parledgroup@notes@endL` .....  
     ..... 2196, 2199, 2227, 2316  
`\parledgroup@notes@endR` .....  
     ..... 2244, 2247, 2271, 2319  
`\parledgroup@notes@startL` .....  
     ..... 964, 2300, 2316  
`\parledgroup@notes@startR` .....  
     ..... 1029, 2300, 2316  
`\parledgroup@notespacing@correction`  
     ..... 2341, 2373–2375  
`\parledgroup@notespacing@correction@accumulated`  
     ..... 2347, 2353, 2374

`\parledgroup@notespacing@correction@modulo`  
     ..... 2348, 2375–2377  
`\parledgroup@notespacing@set@correction`  
     ..... 1902, 2341  
`\parledgroup@series` .....  
     ..... 2298, 2302, 2303,  
     2310, 2311, 2326, 2327, 2333, 2334  
`\parledgroup@type` ..... 2299,  
     2302, 2303, 2310, 2311, 2324, 2331  
`\parledgroupnotespacing` . 2340, 2343  
`\parledgroupseries@` ..... 2297  
`\parledgrouptrue` ..... 10  
`\parledgrouptype@` ..... 2297  
`\pausenumbering` ..... 784  
`\pausenumberingR` ..... 100, 784  
`\pend` ..... 7, 768, 789, 825, 1588  
`\pendL` ..... 768, 899  
`\pendR` ..... 789, 869, 920  
`\prev@abslineverse` .....  
     ..... 2161–2163, 2172–2174  
`\prev@nopbR` ..... 2287  
`\prev@pbr` ..... 2287  
`\prevgraf` .....  
     . 907, 928, 1727, 1728, 1911, 1912  
`\print@columnseparator` .. 1768, 1805  
`\print@eledsectionL` .....  
     ..... 989, 995, 1749, 1955  
`\print@eledsectionR` . 1057, 1760, 1986  
`\print@lineL` ..... 979, 989  
`\print@lineR` ..... 1044, 1057  
`\printlines` ..... 1384  
`\printlinesR` ..... 9, 1373  
`\ProcessOptions` ..... 11  
`\protected@csxdef` ..... 1514, 1525  
`\protected@edef` ..... 850, 893  
`\protected@write` ... 1399, 1411, 1672  
`\ProvidesPackage` ..... 3  
`\pst@rtedLfalse` ..... 33  
`\pst@rtedLtrue` ..... 119, 831  
`\pst@rtedRfalse` ..... 35, 43, 71  
`\pst@rtedRtrue` ..... 104, 127, 875  
`\pstart` ..... 7, 22, 26, 766, 788, 1590  
`\pstartL` ..... 766, 804  
`\pstartnumfalse` ..... 1321, 1326  
`\pstartnumRfalse` ..... 1333, 1338  
`\pstartnumRtrue` .... 1329, 1736, 2265  
`\pstartnumtrue` ..... 1735, 2221  
`\pstartR` ..... 788, 804

**R**

`\Rcolwidth` ..... 5,  
     7, 16, 750, 891, 1038, 1050, 1760  
`\read@linelist` ..... 240, 579  
`\rem@inder` ..... 1243, 1245–1247  
`\resetprevline@` 1785, 1789, 2211, 2259  
`\resumenumbering` ..... 785  
`\resumenumberingR` ..... 100, 785  
`\rightlinenumR` ..... 211, 1256, 1264  
`\rightpstartnumL` ..... 1274  
`\rightpstartnumR` ..... 1274  
`Rightside` (environment) ..... 7, 780  
`\Rightsidehook` ..... 775, 793  
`\Rightsidehookend` ..... 775, 798  
`\rlap` 1256, 1264, 1281, 1289, 1302, 1310  
`\Rlineflag` 9, 206, 218, 1376, 1380, 1421  
`\rule` ..... 1845

**S**

`\savebox` ..... 1749, 1760  
`\sc@n@list` ..... 1244, 1246  
`\secdef` ..... 759  
`\section@num` ..... 120–122  
`\section@numR` ..... 31,  
     45, 56, 57, 63, 64, 105–107, 128–130  
`\select@language` ... 1670, 1672–1675  
`\selectlanguage` ..... 1678  
`\set@line` ..... 671, 694, 715  
`\set@line@action` .....  
     ..... 277, 382, 391, 398, 421, 543  
`\setl@dlp@rbox` ..... 1463, 1478, 1480  
`\setl@drp@rbox` ..... 1465, 1473, 1485  
`\setline` ..... 636  
`\setlinenum` ..... 644  
`\setnamebox` ..... 839, 883, 1609  
`\setprintlines` ..... 1374  
`\shiftedpstartsfalse` ..... 7  
`\shiftedpstartstrue` ..... 6, 8, 9  
`\shiftedversesfalse` ..... 7  
`\shiftedversestrue` ..... 6  
`\showlemma` ..... 680, 703  
`\sidenote@margin` ..... 1441, 1444  
`\sidenote@marginR` ..... 1438, 1468  
`\sidenotecontent@` .....  
     . 1447, 1452, 1453, 1463, 1465,  
     1473, 1474, 1478, 1480, 1481, 1485  
`\sidenotemargin` ..... 1438  
`\sidenotemargin*` ..... 1438  
`\sidenotesep` ..... 1453  
`\skip` ..... 2326, 2333

`\skip@lockoff` ..... 486, 494  
`\skipnumbering` ..... 10, 661  
`\skipnumbering@reg` ..... 665  
`\smash` ..... 1845  
`\splitbotmarks` ..... 2323,  
     2324, 2326, 2327, 2331, 2333, 2334  
`\splitfirstmarks` .....  
     964, 1029, 2302, 2303, 2310, 2311  
`\splittopskip` ..... 959, 1024  
`\stanza@count` ..... 1570, 1584, 1596  
`\stanza@hang` ..... 1572, 1604  
`\stanza@modulo` ..... 1570, 1599  
`\stanzaindentbase` .....  
     ..... 1550, 1560, 1597, 1600  
`\startlock` ..... 652  
`\startstanzahook` ..... 1568  
`\startsub` ..... 619  
`\sub@action` ..... 293, 442, 542  
`\sub@change` ..... 55, 287, 288, 294  
`\sub@lock` ..... 1081  
`\sub@lockR` ..... 52, 302, 304, 306,  
     309, 460, 466, 467, 469, 470,  
     500, 501, 503, 1063, 1119, 1121,  
     1122, 1124, 1180, 1220, 1222, 1224  
`\sub@off` ..... 627, 628  
`\sub@on` ..... 621, 622  
`\subline@num` ..... 220, 343, 361,  
     362, 364, 394, 434, 1082, 1087, 1434  
`\subline@numR` ..... 221,  
     225, 310, 314, 333, 355, 356,  
     358, 387, 425, 548, 552, 1064,  
     1069, 1140, 1147, 1234, 1235, 1430  
`\sublinenumincrement` ..... 7, 182  
`\sublinenumincrement*` ..... 7, 182  
`\sublinenumr@p` . 1377, 1381, 1430, 1434  
`\sublinenumrepR` ..... 208, 221  
`\sublines@false` ..... 53, 291, 1109  
`\sublines@true` ..... 289, 1107  
`\sublock@disp` ..... 1182, 1186, 1191  
`\symplinenum` ..... 1376  
`\sza@penalty` ..... 1579, 1583

**T**

`\temp@` ..... 999, 1000  
`\temp@spacing` ..... 2343, 2344  
`\textwidth` ..... 17, 19, 749, 750  
`\theledlanguageL` ... 1678, 1741, 1947  
`\theledlanguageR` ... 1678, 1752, 1978  
`\thepage` ..... 600, 602, 1400, 1412  
`\thepstart` ..... 767, 787

<code>\thepstartL</code> .....		<b>W</b>	
10, 767, 807, 844, 851, 1320, 1325		<code>\W</code> .....	2286
<code>\thepstartR</code> .....		<code>\wd</code> .....	993
10, 787, 810, 887, 894, 1332, 1337		<code>\WithSuffix</code> .....	202–205, 1438
<code>\thr@@</code> .. 469, 478, 501, 508, 1114, 1122		<code>\writtenlinesLfalse</code> .....	1917, 2206
<code>\topskip</code> .....	2035	<code>\writtenlinesLtrue</code> .....	2203
		<code>\writtenlinesRfalse</code> .....	1918, 2254
		<code>\writtenlinesRtrue</code> .....	2251
<b>U</b>		<b>X</b>	
<code>\unhbox</code> .....	1614,	<code>\x@lemma</code> .....	682–684, 705–707
1764, 1769, 1959, 1962, 1990, 1993		<code>\xifinlist</code> .....	977,
<code>\unhnamebox</code> .....	1609	1000, 1042, 1743, 1754, 1954, 1985	
<code>\unvbox</code> .....	965, 1030, 1616	<code>\xifinlistcs</code> .....	1830–
<code>\unvnamebox</code> .....	1609	1833, 1836–1839, 2136, 2137,	
<code>\usebox</code> .....	1766, 1771	2141, 2142, 2148, 2149, 2153, 2154	
<code>\usenamecount</code> 1571, 1578, 1620, 1652,		<code>\xpg@main@language</code> .....	1710, 1711
1926, 2195, 2198, 2222, 2224,		<code>\xpg@set@language</code> .....	1709
2243, 2246, 2266, 2268, 2301, 2309		<code>\xright@appenditem</code> .....	
<b>V</b>		..... 415, 416, 418, 419,	
<code>\value</code> .....	828, 872, 1595,	423, 430, 432, 439, 444, 446,	
1720, 1721, 1900, 1901, 2208, 2256		448, 451, 453, 455, 463, 465,	
<code>\vbadness</code> .....	958, 1023	474, 497, 499, 506, 525, 526,	
<code>\vbfnoteX</code> .....	1515, 1526	536, 550, 562, 566, 570, 573,	
<code>\vbox</code> .....	839, 883, 1749, 1760	1429, 1433, 1495, 1499, 1515, 1526	
<code>\vl@dbfnote</code> .....	1495, 1499	<b>Z</b>	
<code>\vsplit</code> .....	962, 1027	<code>\zz@@@</code> .....	1392, 1404

## Change History

v0.1		0.15). .....	42
General: First public release .....	1	Lineation can be by pstart (like	
v0.10		in eledmac 0.15). .....	18
General: <code>\edlabel</code> commands on		New management of hang-	
the right side are now correctly		ingsymbol insertion, preventing	
indicated. ....	1	undesirable insertions. ....	57
<code>\edlabel</code> commands which start		Prevent shift of column separator	
a paragraph are now put in the		when a verse is changed .....	57
right place. ....	1	<code>\affixline@numR:</code> Changed	
v0.11		<code>\affixline@numR</code> to allow to	
General: Change <code>\do@lineL</code> and		disable line numbering (like in	
<code>\do@lineR</code> to allow line num-		eledmac 0.15). ....	47
bering by pstart (like in eledmac		<code>\Columns:</code> Line numbering by	

pstart. ....	65	\ledsavedprintlines: Simplified \printlinesR by using \setprintlines .....	53
\get@nextboxR: Change \get@nextboxL and \get@nextboxR to allow to disable line numbering (like in eledmac 0.15). ....	76	\ledstrutR: Added \ledstrutL and \ledstrutR .....	71
Pstart number can be printed in side .....	76	\normalbfnoteX: Removed extraneous spaces from \normalbfnoteX .....	56
v0.12		\Pages: Added \ledstrutL to \Pages .....	69
General: New new management of hangingsymbol insertion, preventing undesirable insertions. ....	57	Added \ledstrutR to \Pages .	70
v0.2		\Rightsidehookend: Added \Leftsidehook, \Leftsidehookend, \Rightsidehook and \Rightsidehookend .....	37
General: Added section of babel related code .....	61	\sublinenumrepR: Added \linenumrepR and \sublinenumrepR .....	21
Fix babel problems .....	1		
\Columns: Added \l@dchecklang and \l@duselanguage to \Columns .....	64		
\Pages: Added \l@duselanguage to \Pages .....	69		
v0.3		v0.3.a	
General: Added \do@lineLhook and \do@lineRhook .....	43	General: Minor \linenummargin fix .....	1
Reorganize for ledarab .....	1	v0.3.b	
\affixline@numR: Changed \affixline@numR to match new eledmac .....	47	General: Improved parallel page balancing .....	1
\do@actions@nextR: Used \do@actions@fixedcode in \do@actionsR .....	45	v0.3.c	
\do@lineL: Added \do@lineLhook to \do@lineL .....	42	General: Compatibilty with Polyglossia .....	1
Simplified \do@lineL by using macros for some common code	42	v0.3a	
\do@lineR: Changed \do@lineR similarly to \do@lineL .....	43	\line@marginR: Don't just set \line@marginR in \linenummargin .....	19
Leftside: Added hooks into Leftside environment .....	36	v0.3b	
\flag@end: Removed extraneous spaces from \flag@end .....	32	\Pages: Added \l@dminpagelines calculation for succeeding page pairs .....	71
\ifledRcol: Moved \ifl@pairing to eledmac .....	15	v0.4	
\ifpst@rtedR: Moved \ifpst@rtedL to eledmac .....	16	General: No more ledparpatch. All patches are now in the main file. ....	1
\l@dlinenumR: Simplified \leftlinenumR and \rightlinenumR by introducing \l@dlinenumR	21	v0.5	
\l@dnumpstartsR: Moved \l@dnumpstartsL to eledmac .	60	General: Corrections about \section and other titles in numbered sections .....	1
		v0.6	
		General: Be able to use \chapter in parallel pages. ....	1
		v0.7	
		General: Option 'shiftedverses' which make there is no blank	

between two parallel verses with inequal length. . . . .	1	v1.1.2	\affixside@noteR: Remove spuri- ous space between line number and line content . . . . .	55
v0.8		v1.12.0		
General: Possibility to have a sym- bol on each hanging of verses, like in the french typogra- phy. Redefine the commande \hangingsymbol to define the character. . . . .	1	General: \beginnumbering is de- fined only on eledmac, not on eledpar. . . . .	16	
v0.9		\Columns: Modify \Columns to en- able to add section's title. . . .	63	
General: Possibility to number \pstart. . . . .	10	Suppress \l@dchecklang from \Columns. . . . .	64	
Possibility to number the pstart with the commands \numberpstarttrue. . . . .	1	\l@dchecklang: Suppress \l@dchecklang which didn't work and was not logical, be- cause both columns could have the same language but not the main language of the docu- ment. . . . .	61	
\ifledRcol: Moved \iflledRcol and \ifnumberingR to eledmac	15	\Pages: Modify \Pages to enable to add section's title. . . . .	68	
v0.9.1		v1.2		
General: The numbering of the pstarts restarts on each \beginnumbering. . . . .	1	General: Support for \led{section} commands in parallel texts. . . .	1	
v0.9.2		v1.2.1		
General: Debug : with \Columns, the hanging indentation now runs on the left columns and the hanging symbol is shown only when \stanza is used. . . . .	1	\initnumbering@sectcountR: For the right section, the counter is defined only once. . . . .	17	
v0.9.3		v1.3		
General: \thepstartL and \thepstartR use now \bfseries and not \bf, which is deprecated and makes con- flicts with memoir class. . . . .	1	General: Manage RTL language. . .	34	
v1.0		v1.3.2		
General: Compatibility with eled- mac. Change name to eledpar. . .	1	General: Debug with some classes. .	1	
Debug in lineation by pstart . .	18	v1.3.3		
v1.0.1		General: Debug on the left notes of the right column. . . . .	55	
General: Correction on \numberonlyfirstinlinenote with lineation by pstart or by page. . . . .	1	\l@dbfnote: Spurious space with footnote in right column. . . .	56	
v1.1		v1.3.4		
General: Shiftedverses becomes shiftedpstarts. . . . .	1	General: Allow to use commands in sidenotes, like it was introduced by eledmac 1.0. . . . .	55	
\pstartR: Add \labelpstarttrue (from eledmac). . . . .	38	v1.3.5		
v1.1.1		\normalbfnoteX: Allows one to redefine \thefootnoteX with alph when some packages are loaded. . . . .	56	
\pstartR: Correct \pstartR bug in- troduced by 1.1. . . . .	38	v1.4		
		General: Added \do@insidelineLhook and \do@insidelineRhook . . .	43	

v1.4.1		and <code>\l@dcnote</code> defined only one time, in <code>eledmac</code> . . . . .	55
	<code>\normalbfnoteX</code> : Fix bug with normal familiar footnotes when mixing RTL and LTR text. . .	Add <code>\beforecolumnseparator</code> and <code>\aftercolumnseparator</code> . .	6
	<code>astanza</code> : Enable the use of <code>stanzaindentrepetition</code> within <code>astanza</code> environment. . . . .	Add <code>\columnspan</code> . . . . .	6
		Add, as in <code>eledmac</code> , new system of sectioning commands. . . . .	1
v1.4.2		Add, as in <code>eledmac</code> , possibility to insert something after <code>\pends</code> / verses. . . . .	1
	<code>\line@list@stuffR</code> : Open / close immediatly the line-list file when in <code>minipage</code> , except if the <code>minipage</code> is a <code>ledgroup</code> . . . . .	Add, as in <code>eledmac</code> , possibility to insert something between <code>\pstarts</code> / verse. . . . .	1
v1.4.3		Change <code>\do@lineR</code> and <code>\do@lineR</code> to allow new sectioning commands. . . . .	42
	General: Corrects a false hanging verse when a verse is exactly the length of a line. . . . .	Compatibility with <code>musixtex</code> . . .	1
	<code>\inserthangingsymbolR</code> : Hang verse is now not automatically flush right. . . . .	Debug <code>eledmac</code> sectioning command after using <code>\resumenumbering</code> . . . . .	1
	<code>\pendL</code> : Spurious spaces in <code>\pendL</code> . 40	New sectioning commands, as in <code>eledmac</code> . . . . .	14
	<code>\pendR</code> : Spurious spaces in <code>\pstartR</code> . . . . .	<code>\pendL</code> : As in <code>eledmac</code> , <code>\pendL</code> can have an optional argument. . .	40
	<code>\pstartR</code> : Spurious spaces in <code>\pstartL</code> and <code>\pstartR</code> . . . .	<code>\pendR</code> : As in <code>eledmac</code> , <code>\pendR</code> can have an optional argument. . .	41
v1.5.0		<code>\print@columnseparator</code> : Move some code of <code>\Columns</code> to <code>\print@columnseparator</code> . . .	65
	General: Add, as in <code>eledmac</code> , features to manage page breaks. . .	<code>\pstartR</code> : As in <code>eledmac</code> , <code>\pendL</code> and <code>\pendR</code> can have an optional argument. . . . .	38
	<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. . . . .	<code>\sidenotemargin*</code> : <code>\sidenotemargin</code> is now directly defined in <code>eledmac</code> to be able to manage <code>eledpar</code> . . . . .	55
v1.6.0		Add <code>\sidenotemargin*</code> . . . . .	55
	General: Add tool and documentation for parallel <code>ledgroups</code> . . .	<code>\theledlanguageR</code> : Correct left/right language setting with <code>polyglossia</code> . . . . .	63
v1.7.0			
	General: Add, as in <code>eledmac</code> , features to make crossrefs with <code>pstart</code> numbers. . . . .		
v1.8.0			
	General: <code>\l@dlsnote</code> , <code>\l@drsnote</code>		