

# 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:  
<https://lists.berlios.de/pipermail/ledmac-users/>

## Contents

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

---

\*This file (**eledpar.dtx**) has version number v1.4.3, last revised 2013/10/27.

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

<sup>‡</sup>maieul at maieul dot net

<b>7 Verse</b>	<b>9</b>
<b>8 Implementation overview</b>	<b>12</b>
<b>9 Preliminaries</b>	<b>12</b>
9.1 Messages . . . . .	13
<b>10 Sectioning commands</b>	<b>13</b>
<b>11 Line counting</b>	<b>16</b>
11.1 Choosing the system of lineation . . . . .	16
11.2 Line-number counters and lists . . . . .	19
11.3 Reading the line-list file . . . . .	20
11.4 Commands within the line-list file . . . . .	21
11.5 Writing to the line-list file . . . . .	29
<b>12 Marking text for notes</b>	<b>31</b>
<b>13 Parallel environments</b>	<b>33</b>
<b>14 Paragraph decomposition and reassembly</b>	<b>35</b>
14.1 Boxes, counters, <code>\pstart</code> and <code>\pend</code> . . . . .	35
14.2 Processing one line . . . . .	39
14.3 Line and page number computation . . . . .	41
14.4 Line number printing . . . . .	43
14.5 Pstart number printing in side . . . . .	46
14.6 Add insertions to the vertical list . . . . .	47
14.7 Penalties . . . . .	48
14.8 Printing leftover notes . . . . .	49
<b>15 Footnotes</b>	<b>49</b>
15.1 Normal footnote formatting . . . . .	49
<b>16 Cross referencing</b>	<b>50</b>
<b>17 Side notes</b>	<b>51</b>
<b>18 Familiar footnotes</b>	<b>53</b>
<b>19 Verse</b>	<b>54</b>
<b>20 Naming macros</b>	<b>56</b>
<b>21 Counts and boxes for parallel texts</b>	<b>57</b>
<b>22 Fixing babel</b>	<b>58</b>
<b>23 Parallel columns</b>	<b>60</b>

<i>List of Figures</i>	3
<b>24 Parallel pages</b>	<b>63</b>
<b>25 The End</b>	<b>72</b>
<b>Appendix A Some things to do when changing version</b>	<b>73</b>
Appendix A.1 Migration to <code>eledpar</code> 1.4.3 . . . . .	73
<b>References</b>	<b>73</b>
<b>Index</b>	<b>73</b>
<b>Change History</b>	<b>81</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 8 through 25); 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 8, unless you are particularly interested in the innards of `eledpar`.

### 2 The `eledpar` package

A file may mix *numbered* and *unnumbered* text. Numbered text is printed with marginal line numbers and can include footnotes and endnotes that are referenced

to those line numbers: this is how you'll want to print the text that you're editing. Unnumbered text is not printed with line numbers, and you can't use `eledmac`'s note commands with it: this is appropriate for introductions and other material added by the editor around the edited text.

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

## 2.1 General

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

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

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

```
\maxchunks{5120}
```

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

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

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

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

### 3 Parallel columns

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

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

```
\begin{pairs}
\begin{Leftside} ... \end{Leftside}
\begin{Rightside} ... \end{Rightside}
\Columns
\begin{Leftside} ... \end{Leftside}
...
\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. 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. When you use `\stanza`, the visible rule may shift when a verse has a hanging indent. To prevent shifting, use `\setstanzaindents` outside the **Leftside** or **Rightside** environment.

### 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 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 likewise in the **Rightside** environment. The number of **Leftside** and **Rightside** environments must be the same.

**\firstlinenum**      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.

**\pstart**      In a serial (non-parallel) mode, each numbered paragraph, or chunk, is con-

**\pend**

---

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

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

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

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

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

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

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

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

## 6 Numbering text lines and paragraphs

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

```
\beginnumbering
<text>
\endnumbering
```

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

The `\beginnumbering` macro resets the line number to zero, reads an auxiliary file called `<jobname>.nn` (where `<jobname>` is the name of the main input file for this job, and `nn` is 1 for the first numbered section, 2 for the second section, and so on), and then creates a new version of this auxiliary file to collect information during this run. Separate auxiliary files are maintained for right hand texts and these are named `<jobname>.nnR`, using the ‘R’ to distinguish them from the left hand and serial (non-parallel) texts.

`\memorydump` The command `\memorydump` effectively performs an `\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 referred to by `\ledsavedprintlines`.



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

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

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

```
\numberpstarttrue
\numberpstartfalse
  \thepstartL
  \thepstartR
```

## 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 `\setstanzaindents` to set the stanza indents.

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

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

```
\newcommand*{\stanzanum}[2][\stanzaindentbase]{%
  \hskip -#1\llap{\textbf{#2}}\hskip #1\ignorespaces}
```

```
\newcommand{\interstanza}{\par\mbox{}\skipnumbering}
```

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

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

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

\end{astanza}
...
```

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 `LATEX` two time. Example for the french typographie

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

## 8 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`.

## 9 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}[2013/10/27 v1.4.3 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 \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.
11 \l@dpairingfalse
12 \newif\ifl@dpaging
13 \l@dpagingfalse
14 \ledRcolfalse

\Lcolwidth The widths of the left and right parallel columns (or pages).
\Rcolwidth 15 \newdimen\Lcolwidth
16 \Lcolwidth=0.45\textwidth
17 \newdimen\Rcolwidth
18 \Rcolwidth=0.45\textwidth
19
```

## 9.1 Messages

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

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

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

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

## 10 Sectioning commands

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

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

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

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

```

32 \pst@rtedLfalse
33 \newif\ifpst@rtedR
34 \pst@rtedRfalse
35

```

`\beginnumbering` For parallel processing the original `\beginnumbering` is extended to zero `\l@dnumpstartsL` — the number of chunks to be processed. It also sets `\ifpst@rtedL` to FALSE.

```

36 \providecommand*\beginnumbering{%
37   \ifnumbering
38     \led@err@NumberingStarted
39   \endnumbering
40 \fi
41 \global\l@dnumpstartsL \z@
42 \global\pst@rtedLfalse
43 \global\numberingtrue
44 \global\advance\section@num \@ne
45 \initnumbering@reg
46 \message{Section \the\section@num}%
47 \line@list@stuff{\jobname.\extensionchars\the\section@num}%
48 \l@dend@stuff}

```

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

```

49 \newcommand*\beginnumberingR{%
50   \ifnumberingR
51     \led@err@NumberingStarted
52   \endnumberingR
53 \fi
54 \global\l@dnumpstartsR \z@
55 \global\pst@rtedRfalse
56 \global\numberingRtrue
57 \global\advance\section@numR \@ne
58 \global\absline@numR \z@
59 \global\line@numR \z@
60 \global\@lockR \z@
61 \global\sub@lockR \z@
62 \global\sublines@false
63 \global\let\next@page@numR\relax
64 \global\let\sub@change\relax
65 \message{Section \the\section@numR R }%
66 \line@list@stuffR{\jobname.\extensionchars\the\section@numR R}%
67 \l@dend@stuff
68 \setcounter{pstartR}{1}
69 \begingroup
70 \initnumbering@sectcmd
71 \initnumbering@sectcountR
72 }

```

73

`\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.

```

74 \def\endnumberingR{%
75   \ifnumberingR
76     \global\numberingRfalse
77     \normal@pars
78     \ifl@dpairing
79       \global\pst@rtedRfalse
80   \else
81     \ifx\insertlines@listR\empty\else
82       \global\noteschanged@true
83     \fi
84     \ifx\line@listR\empty\else
85       \global\noteschanged@true
86     \fi
87   \fi
88   \ifnoteschanged@
89     \led@mess@NotesChanged
90   \fi
91 \else
92   \led@err@NumberingNotStarted
93 \fi\endgroup}
94
```

`\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  $\text{\LaTeX}$  counter in `\numberingR`.

```

95 \newcounter{chapterR}
96 \newcounter{sectionR}
97 \newcounter{subsectionR}
98 \newcounter{subsubsectionR}
99 \newcommand{\initnumbering@sectcountR}{
100 \let\c@chapter\c@chapterR
101 \let\c@section\c@sectionR
102 \let\c@subsection\c@subsectionR
103 \let\c@subsubsection\c@subsubsectionR
104 }
```

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

```

\resumenumberingR 105 \newcommand*{\pausenumberingR}{%
106   \endnumberingR\global\numberingRtrue}
107 \newcommand*{\resumenumberingR}{%
108   \ifnumberingR
109     \global\pst@rtedRtrue
```

```

110 \global\advance\section@numR \@ne
111 \led@mess@SectionContinued{\the\section@numR R}%
112 \line@list@stuffR{\jobname.\extensionchars\the\section@numR R}%
113 \l@dend@stuff
114 \else
115 \led@err@numberingShouldHaveStarted
116 \endnumberingR
117 \beginnumberingR
118 \fi}
119

```

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

```

120 \newcommand*{\memorydumpL}{%
121 \endnumbering
122 \numberingtrue
123 \global\pst@rtedLtrue
124 \global\advance\section@num \@ne
125 \led@mess@SectionContinued{\the\section@num}%
126 \line@list@stuff{\jobname.\extensionchars\the\section@num}%
127 \l@dend@stuff}
128 \newcommand*{\memorydumpR}{%
129 \endnumberingR
130 \numberingRtrue
131 \global\pst@rtedRtrue
132 \global\advance\section@numR \@ne
133 \led@mess@SectionContinued{\the\section@numR R}%
134 \line@list@stuffR{\jobname.\extensionchars\the\section@numR R}%
135 \l@dend@stuff}
136

```

## 11 Line counting

### 11.1 Choosing the system of lineation

M 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.

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

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



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

```
137 \newif\ifbypage@R
138 \newif\ifbypstart@R
139 \bypage@Rfalse
140 \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`.

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

`\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.

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

```

173   \fi
174   \fi}}

```

By default put right text numbers at the right.

```

175 \line@marginR=\@ne
176

```

`\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.

```

177 \newcounter{firstlinenumR}
178 \setcounter{firstlinenumR}{5}
179 \newcounter{linenumincrementR}
180 \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.

```

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

```

`\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 `\provide`ing them.

```

\linenumincrement
\firstsublinenum
\sublinenumincrement
186 \providecommand*\firstlinenum{}
187 \providecommand*\linenumincrement{}
188 \providecommand*\firstsublinenum{}
189 \providecommand*\sublinenumincrement{}
190 \renewcommand*\firstlinenum[1]{%
191   \ifledRcol \setcounter{firstlinenumR}{#1}%
192   \else      \setcounter{firstlinenumR}{#1}%
193   \fi}
194 \renewcommand*\linenumincrement[1]{%
195   \ifledRcol \setcounter{linenumincrementR}{#1}%
196   \else      \setcounter{linenumincrementR}{#1}%
197   \fi}
198 \renewcommand*\firstsublinenum[1]{%
199   \ifledRcol \setcounter{firstsublinenumR}{#1}%
200   \else      \setcounter{firstsublinenumR}{#1}%
201   \fi}
202 \renewcommand*\sublinenumincrement[1]{%
203   \ifledRcol \setcounter{sublinenumincrementR}{#1}%
204   \else      \setcounter{sublinenumincrementR}{#1}%
205   \fi}
206

```

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

```

207 \newcommand*\Rlineflag{\R}
208
\linenumrepR \linenumrepR{<ctr>} typesets the right line number <ctr>, and similarly \sublinenumrepR
\sublinenumrepR for subline numbers.
209 \newcommand*\linenumrepR[1]{\@arabic{#1}}
210 \newcommand*\sublinenumrepR[1]{\@arabic{#1}}
211
\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.
212 \newcommand*\leftlinenumR{%
213 \l@dlinenumR
214 \kern\linenumsep}
215 \newcommand*\rightlinenumR{%
216 \kern\linenumsep
217 \l@dlinenumR}
218 \newcommand*\l@dlinenumR{%
219 \numlabfont\linenumrepR{\line@numR}\Rlineflag%
220 \ifsublines@
221 \ifnum\subline@num>\z@
222 \unskip\fullstop\sublinenumrepR{\subline@numR}%
223 \fi
224 \fi}
225

```

## 11.2 Line-number counters and lists

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

```

226 \newcount\line@numR
227 \newcount\subline@numR
228 \newcount\absline@numR
229
\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:

```

```

230 \list@create{\line@listR}
231 \list@create{\insertlines@listR}
232 \list@create{\actionlines@listR}
233 \list@create{\actions@listR}
234

```

`\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.

```

235 \list@create{\linesinpar@listL}
236 \list@create{\linesinpar@listR}
237 \list@create{\maxlinesinpar@list}
238

```

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

```

239 \newcount\page@numR
240

```

### 11.3 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.

```

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

```

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

```

242 \ifledRcol
243 \list@clear{\line@listR}%
244 \list@clear{\insertlines@listR}%
245 \list@clear{\actionlines@listR}%
246 \list@clear{\actions@listR}%
247 \list@clear{\linesinpar@listR}%
248 \list@clear{\linesonpage@listR}
249 \else
250 \list@clearing@reg
251 \list@clear{\linesinpar@listL}%
252 \list@clear{\linesonpage@listL}%
253 \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).

```

254 \list@clear{\maxlinesinpar@list}

```

Now get the file and interpret it.

```

255 \get@linelistfile{#1}%
256 \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.

```

257 \ifledRcol
258   \global\page@numR=\m@ne
259   \ifx\actionlines@listR\empty
260     \gdef\next@actionlineR{1000000}%
261   \else
262     \gl@p\actionlines@listR\to\next@actionlineR
263     \gl@p\actions@listR\to\next@actionR
264   \fi
265 \else
266   \global\page@num=\m@ne
267   \ifx\actionlines@list\empty
268     \gdef\next@actionline{1000000}%
269   \else
270     \gl@p\actionlines@list\to\next@actionline
271     \gl@p\actions@list\to\next@action
272   \fi
273 \fi}
274

```

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.

## 11.4 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.

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

```

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

```

```

289 \ifnum\sub@change>\z@
290 \sublines@true
291 \else
292 \sublines@false
293 \fi
294 \sub@action
295 \let\sub@change\relax
296 \fi
297 \ifcase\@lockR
298 \or
299 \@lockR \tw@
300 \or\or
301 \@lockR \z@
302 \fi
303 \ifcase\sub@lockR
304 \or
305 \sub@lockR \tw@
306 \or\or
307 \sub@lockR \z@
308 \fi
309 \ifsublines@
310 \ifnum\sub@lockR<\tw@
311 \advance\subline@numR \@ne
312 \fi
313 \else
314 \ifnum\@lockR<\tw@
315 \advance\line@numR \@ne \subline@numR \z@
316 \fi
317 \fi}
318
319 \renewcommand*{\@l}[2]{%
320 \fix@page{#1}%
321 \ifledRcol
322 \@l@regR
323 \else
324 \@l@reg
325 \fi}
326

```

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

```

\fix@page 327 \newcount\last@page@numR
328 \last@page@numR=-10000
329 \renewcommand*{\fix@page}[1]{%
330 \ifledRcol
331 \ifnum #1=\last@page@numR
332 \else
333 \ifbypage@R
334 \line@numR \z@ \subline@numR \z@
335 \fi
336 \page@numR=#1\relax

```

```

337     \last@page@numR=#1\relax
338     \def\next@page@numR{#1}%
339     \fi
340 \else
341     \ifnum #1=\last@page@num
342     \else
343         \ifbypage@
344             \line@num \z@ \subline@num \z@
345         \fi
346         \page@num=#1\relax
347         \last@page@num=#1\relax
348         \def\next@page@num{#1}%
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@dtmpcnta=-\subline@numR
426     \else
427       \@l@dtmpcnta=-\line@numR
428     \fi
429     \advance\@l@dtmpcnta by -5000\relax
430     \xright@appenditem{\the\@l@dtmpcnta}\to\actions@listR
431   \else
432     \xright@appenditem{\the\absline@num}\to\actionlines@list
433     \ifsublines@
434       \@l@dtmpcnta=-\subline@num
435     \else
436       \@l@dtmpcnta=-\line@num
437     \fi
438     \advance\@l@dtmpcnta by -5000\relax
439     \xright@appenditem{\the\@l@dtmpcnta}\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 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 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 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

```

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

```

599 \newcommand*{\new@lineR}{%
600   \write\linenum@outR{\string\@l[\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.

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

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

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

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

630 \renewcommand*{\advanceline}[1]{%
631   \ifledRcol \write\linenum@outR{\string\@adv[#1]}%
632   \else      \write\linenum@out{\string\@adv[#1]}%
633   \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.

634 \renewcommand*{\setline}[1]{%
635   \ifnum#1<\z@

```

```

636     \led@warn@BadSetline
637 \else
638     \ifledRcol \write\linenum@outR{\string\@set[#1]}%
639     \else      \write\linenum@out{\string\@set[#1]}%
640     \fi
641 \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.

```

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

```

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

**\endlock**

```

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

```

**\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.

```

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

```

## 12 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.

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

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

```
689 \renewcommand{\edtext}[2]{\leavevmode
690   \begingroup%
691     \renewcommand{\@tag}{\no@expands #1}%
692     \set@line%
693     \ifl@rcol \global\insert@countR \z@%
694     \else      \global\insert@count \z@ \fi%
695     \ignorespaces #2\relax%
696     \@ifundefined{xpg@main@language}{%if not polyglossia
697       \flag@start}%
698       {\if@RTL\flag@end\else\flag@start\fi% be careful on the direction of writing with p
```



```

699     }%
700   \endgroup%
701   \showlemma{#1}%
702   \ifx\end@lemmas\empty \else%
703     \glp\end@lemmas\to\x@lemma%
704     \x@lemma%
705     \global\let\x@lemma=\relax%
706   \fi%
707   \@ifundefined{xpg@main@language}{%if not polyglossia
708     \flag@end}%
709     {\ifRTL\flag@start\else\flag@end\fi% be careful on the direction of writing with polyglossia
710     }%
711 }
712

```

`\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`.

```

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

```

## 13 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 734 \newenvironment{pairs}{%}
735   \l@d@pairingtrue
736   \l@d@pagingfalse

```

```

737 }{%
738   \l@dpairingfalse
739 }

```

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.

```

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

```

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

```

757 \newif\ifinstanzaL
758 \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.

```

759 \newenvironment{Leftside}{%
760   \ledRcolfalse
761   \let\beginnumbering\beginnumbering\setcounter{pstartL}{1}
762   \let\pstart\pstartL
763   \let\thepstart\thepstartL
764   \let\pend\pendL
765   \let\memorydump\memorydumpL
766   \Leftsidehook
767   \let\oldstanza\stanza
768   \renewcommand{\stanza}{\oldstanza\global\instanzaLtrue}
769 }{
770   \let\stanza\oldstanza
771   \Leftsidehookend}

```

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

`\Rightsidehook` 772 `\newcommand*{\Leftsidehook}{}`  
`\Rightsidehookend` 773 `\newcommand*{\Leftsidehookend}{}`  
774 `\newcommand*{\Rightsidehook}{}`  
775 `\newcommand*{\Rightsidehookend}{}`  
776

`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.

```

777 \newenvironment{Rightside}{%
778   \ledRcoltrue
779   \let\beginnumbering\beginnumberingR
780   \let\endnumbering\endnumberingR
781   \let\pausenumbering\pausenumberingR
782   \let\resumenumbering\resumenumberingR
783   \let\memorydump\memorydumpR
784   \let\thepstart\thepstartR
785   \let\pstart\pstartR
786   \let\pend\pendR
787   \let\lineation\lineationR
788   \Rightsidehook
789   \let\oldstanza\stanza
790   \renewcommand{\stanza}{\oldstanza\global\instanzaRtrue}
791 }{%
792   \ledRcolfalse
793   \let\stanza\oldstanza
794   \Rightsidehookend
795 }
796
```

## 14 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.

### 14.1 Boxes, counters, `\pstart` and `\pend`

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

```
797 \newcount\num@linesR
798 \newbox\one@lineR
799 \newcount\par@lineR
```

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

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

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

```
800
801 \newcounter{pstartL}
802 \newcounter{pstartLold}
803 \renewcommand{\thepstartL}{\bfseries\@arabic\c@pstartL}. }
804 \newcounter{pstartR}
805 \newcounter{pstartRold}
806 \renewcommand{\thepstartR}{\bfseries\@arabic\c@pstartR}. }
807
808 \newcommand*{\pstartL}{%
809 \if@nobreak%
810   \let\@oldnobreak\@nobreaktrue%
811 \else%
812   \let\@oldnobreak\@nobreakfalse%
813 \fi%
814   \@nobreaktrue%
815   \ifnumbering \else%
816     \led@err@PstartNotNumbered%
817     \beginnumbering%
818   \fi%
819   \ifnumberedpar@%
820     \led@err@PstartInPstart%
821   \pend%
822 \fi%
```

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

```
823 \ifpstart@rtedL\else%
824   \setcounter{pstartLold}{\value{pstartL}}%
```

```

825 \list@clear{\inserts@list}%
826 \global\let\next@insert=\empty%
827 \global\pst@rtedLtrue%
828 \fi%
829 \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.

```

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

849 \newcommand*{\pstartR}{%
850 \if@nobreak%
851 \let\@oldnobreak\@nobreaktrue%
852 \else%
853 \let\@oldnobreak\@nobreakfalse%
854 \fi%
855 \@nobreaktrue%
856 \ifnumberingR \else%
857 \led@err@PstartNotNumbered%
858 \beginnumberingR%
859 \fi%
860 \ifnumberedpar@%
861 \led@err@PstartInPstart%
862 \pendR%
863 \fi%
864 \ifpst@rtedR\else%
865 \setcounter{pstartRold}{\value{pstartR}}%
866 \list@clear{\inserts@listR}%
867 \global\let\next@insertR=\empty%
868 \global\pst@rtedRtrue%
869 \fi%
870 \begingroup\normal@pars%
871 \global\advance\l@dnumpstartsR \@ne%

```

```

872 \ifnum\l@dnumpstartsR>\l@dc@maxchunks%
873   \led@err@TooManyPstarts%
874   \global\l@dnumpstartsR=\l@dc@maxchunks%
875 \fi%
876 \global\setnamebox{l@dRcolrawbox\the\l@dnumpstartsR}=\vbox\bgroup%
877   \ifautopar\else%
878     \ifnumberpstart%
879       \ifsidepstartnum\else%
880         \thepstartR%
881       \fi%
882     \fi%
883   \fi%
884 \hsize=\Rcolwidth%
885 \numberedpar@true%
886 \iflabelpstart\protected@edef\@currentlabel%
887   {\p@pstartR\thepstartR}\fi%
888 }

```

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

```

889 \newcommand*{\pendL}{\ifnumbering \else%
890   \led@err@PendNotNumbered%
891 \fi%
892 \ifnumberedpar@ \else%
893   \led@err@PendNoPstart%
894 \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.

```

895 \l@dzeropenalties%
896 \endgraf\global\num@lines=\prevgraf\egroup%
897 \global\par@line=0%

```

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

```

898 \endgroup%
899 \ignorespaces%
900 \@oldnobreak%
901 \ifnumberpstart%
902   \addtocounter{pstartL}{1}%
903 \fi}
904

```

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

```

905 \newcommand*{\pendR}{\ifnumberingR \else%
906   \led@err@PendNotNumbered%
907 \fi%
908 \ifnumberedpar@ \else%

```

```

909 \led@err@PendNoPstart%
910 \fi%
911 \l@dzeropenalties%
912 \endgraf\global\num@linesR=\prevgraf\egroup%
913 \global\par@lineR=0%
914 \endgroup%
915 \ignorespaces%
916 \@oldnobreak%
917 \ifnumberpstart%
918 \addtocounter{pstartR}{1}%
919 \fi%
920 }
921

```

## 14.2 Processing one line

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

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

```

922 \newbox\l@dleftbox
923 \newbox\l@drightbox
924

```

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

```

\countRline 925 \newcount\countLline
926 \countLline \z@
927 \newcount\countRline
928 \countRline \z@
929

```

`\@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 930 \newcount\@donereallinesL
931 \newcount\@donetotallinesL
932 \newcount\@donereallinesR
933 \newcount\@donetotallinesR
934

```

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

```

935 \newcommand*\do@lineL{%
936 \advance\countLline \@ne
937 \ifvbox\namebox{\l@dLcolrawbox\the\l@dpscl}%
938 {\vbadness=10000
939 \splittopskip=\z@

```

```

940 \do@lineLhook
941 \l@emptyd@ta
942 \global\setbox\one@line=\vsplit\namebox{l@dLcolrawbox\the\l@dpscL}
943 to\baselineskip}%
944 \unvbox\one@line \global\setbox\one@line=\lastbox
945 \getline@numL
946 \ifnum\@lock>\@ne\inserthangingsymboltrue\else\inserthangingsymbolfalse\fi
947 \setbox\l@dleftbox
948 \hb@xt@ \Lcolwidth{%
949 \affixpstart@numL
950 \affixline@num
951 \l@dld@ta
952 \add@inserts
953 \affixside@note
954 \l@dlsn@te
955 {\ledllfill\hb@xt@ \wd\one@line{\do@insidelineLhook\inserthangingsymbolL\new@line\l@d
956 \l@drsn@te
957 }}%
958 \add@penaltiesL
959 \global\advance\@donereallinesL\@ne
960 \global\advance\@donetotallinesL\@ne
961 \else
962 \setbox\l@dleftbox \hb@xt@ \Lcolwidth{\hspace*\Lcolwidth}}%
963 \global\advance\@donetotallinesL\@ne
964 \fi}
965
966

```

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

```

\do@lineRhook 967 \newcommand*{\do@lineLhook}{%
\do@insidelineLhook 968 \newcommand*{\do@lineRhook}{%
\do@insidelineRhook 969 \newcommand*{\do@insidelineLhook}{%
970 \newcommand*{\do@insidelineRhook}{%
971

```

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

```

972 \newcommand*{\do@lineR}{%
973 \advance\countRline \@ne
974 \ifvbox\namebox{l@dRcolrawbox\the\l@dpscR}%
975 {\vbadness=10000
976 \splittopskip=\z@
977 \do@lineRhook
978 \l@emptyd@ta
979 \global\setbox\one@lineR=\vsplit\namebox{l@dRcolrawbox\the\l@dpscR}
980 to\baselineskip}%
981 \unvbox\one@lineR \global\setbox\one@lineR=\lastbox
982 \getline@numR
983 \ifnum\@lockR>\@ne\inserthangingsymbolRtrue\else\inserthangingsymbolRfalse\fi

```



```

984 \setbox\l@drightbox
985 \hb@xt@ \Rcolwidth{%
986     \affixstart@numR
987     \affixline@numR
988     \l@dld@ta
989     \add@insertsR
990     \affixside@noteR
991     \l@dlsn@te
992     {\correcthangingR\ledllfill\hb@xt@ \wd\one@lineR{\do@insidelineRhook\inserthangingsymbolR\new@lin
993     \l@drsn@te
994 }}%
995 \add@penaltiesR
996 \global\advance\@donereallinesR\@ne
997 \global\advance\@donetotallinesR\@ne
998 \else
999 \setbox\l@drightbox \hb@xt@ \Rcolwidth{\hspace*{\Rcolwidth}}
1000 \global\advance\@donetotallinesR\@ne
1001 \fi}
1002
1003

```

### 14.3 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.

```

1004 \newcommand*{\getline@numR}{%
1005     \global\advance\absline@numR \@ne
1006     \do@actionsR
1007     \do@ballastR
1008 \ifnumberline
1009     \ifsublines@
1010         \ifnum\sub@lockR<\tw@
1011             \global\advance\subline@numR \@ne
1012         \fi
1013     \else
1014         \ifnum\@lockR<\tw@
1015             \global\advance\line@numR \@ne
1016             \global\subline@numR \z@
1017         \fi
1018     \fi
1019 \fi
1020 }
1021 \newcommand*{\getline@numL}{%
1022     \global\advance\absline@num \@ne
1023     \do@actions
1024     \do@ballast
1025 \ifnumberline
1026     \ifsublines@
1027         \ifnum\sub@lock<\tw@

```

```

1028     \global\advance\subline@num \@ne
1029     \fi
1030   \else
1031     \ifnum\@lock<\tw@
1032       \global\advance\line@num \@ne
1033       \global\subline@num \z@
1034     \fi
1035   \fi
1036 \fi
1037 }
1038
1039

```

`\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.

```

1040 \newcommand*{\do@ballastR}{\global\ballast@count=\z@
1041   \begingroup
1042     \advance\absline@numR \@ne
1043     \ifnum\next@actionlineR=\absline@numR
1044       \ifnum\next@actionR>-1001
1045         \global\advance\ballast@count by -\c@ballast
1046       \fi
1047     \fi
1048   \endgroup}

```

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

`\do@actions@nextR`

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

```

1049 \newcommand*{\do@actions@fixedcodeR}{%
1050   \ifcase\@l@dtmpcnta%
1051     \or% % 1001
1052       \global\sublines@true
1053     \or% % 1002
1054       \global\sublines@false
1055     \or% % 1003
1056       \global\@lockR=\@ne
1057     \or% % 1004
1058       \ifnum\@lockR=\tw@
1059         \global\@lockR=\thr@@
1060       \else
1061         \global\@lockR=\z@
1062       \fi
1063     \or% % 1005
1064       \global\sub@lockR=\@ne
1065     \or% % 1006
1066       \ifnum\sub@lockR=\tw@
1067         \global\sub@lockR=\thr@@

```

```

1068 \else
1069 \global\sub@lockR=\z@
1070 \fi
1071 \or% % 1007
1072 \l@dskipnumbertrue
1073 \else
1074 \led@warn@BadAction
1075 \fi}
1076
1077
1078 \newcommand*{\do@actionsR}{%
1079 \global\let\do@actions@nextR=\relax
1080 \@l@dttempcntb=\absline@numR
1081 \ifnum\@l@dttempcntb<\next@actionlineR\else
1082 \ifnum\next@actionR>-1001\relax
1083 \global\page@numR=\next@actionR
1084 \ifbypage@R
1085 \global\line@numR \z@ \global\subline@numR \z@
1086 \fi
1087 \else
1088 \ifnum\next@actionR<-4999\relax % 9/05 added relax here
1089 \@l@dttempcnta=-\next@actionR
1090 \advance\@l@dttempcnta by -5001\relax
1091 \ifsublines@
1092 \global\subline@numR=\@l@dttempcnta
1093 \else
1094 \global\line@numR=\@l@dttempcnta
1095 \fi
1096 \else
1097 \@l@dttempcnta=-\next@actionR
1098 \advance\@l@dttempcnta by -1000\relax
1099 \do@actions@fixedcodeR
1100 \fi
1101 \fi
1102 \ifx\actionlines@listR\empty
1103 \gdef\next@actionlineR{1000000}%
1104 \else
1105 \gl@p\actionlines@listR\to\next@actionlineR
1106 \gl@p\actions@listR\to\next@actionR
1107 \global\let\do@actions@nextR=\do@actionsR
1108 \fi
1109 \fi
1110 \do@actions@nextR}
1111

```

## 14.4 Line number printing

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

```

\ch@cksub@l@ckR 1112
\ch@ck@l@ckR
\fx@l@cksR
\affixline@numR

```

```

1113 \providecommand*{\l@dcalcnum}[3]{%
1114   \ifnum #1 > #2\relax
1115     \l@dttempcnta = #1\relax
1116     \advance\l@dttempcnta by -#2\relax
1117     \divide\l@dttempcnta by #3\relax
1118     \multiply\l@dttempcnta by #3\relax
1119     \advance\l@dttempcnta by #2\relax
1120   \else
1121     \l@dttempcnta=#2\relax
1122   \fi}
1123
1124 \newcommand*{\ch@cksub@l@ckR}{%
1125   \ifcase\sub@lockR
1126   \or
1127     \ifnum\sublock@disp=\@ne
1128       \l@dttempcntb \z@ \l@dttempcnta \@ne
1129     \fi
1130   \or
1131     \ifnum\sublock@disp=\tw@
1132     \else
1133       \l@dttempcntb \z@ \l@dttempcnta \@ne
1134     \fi
1135   \or
1136     \ifnum\sublock@disp=\z@
1137       \l@dttempcntb \z@ \l@dttempcnta \@ne
1138     \fi
1139   \fi}
1140
1141 \newcommand*{\ch@ck@l@ckR}{%
1142   \ifcase\@lockR
1143   \or
1144     \ifnum\lock@disp=\@ne
1145       \l@dttempcntb \z@ \l@dttempcnta \@ne
1146     \fi
1147   \or
1148     \ifnum\lock@disp=\tw@
1149     \else
1150       \l@dttempcntb \z@ \l@dttempcnta \@ne
1151     \fi
1152   \or
1153     \ifnum\lock@disp=\z@
1154       \l@dttempcntb \z@ \l@dttempcnta \@ne
1155     \fi
1156   \fi}
1157
1158 \newcommand*{\f@x@l@cksR}{%
1159   \ifcase\@lockR
1160   \or
1161     \global\@lockR \tw@
1162   \or \or

```

```

1163   \global\@lockR \z@
1164   \fi
1165   \ifcase\sub@lockR
1166   \or
1167     \global\sub@lockR \tw@
1168   \or \or
1169     \global\sub@lockR \z@
1170   \fi}
1171
1172
1173 \newcommand*{\affixline@numR}{%
1174 \ifnumberline
1175 \ifl@dskipnumber
1176   \global\l@dskipnumberfalse
1177 \else
1178   \ifsublines@
1179     \@l@tempcntb=\subline@numR
1180     \l@dcalcnum{\subline@numR}{\c@firstsublinenumR}{\c@sublinenumincrementR}%
1181     \ch@cksub@lockR
1182   \else
1183     \@l@tempcntb=\line@numR
1184     \ifx\linenumberlist\empty
1185       \l@dcalcnum{\line@numR}{\c@firstlinenumR}{\c@linenumincrementR}%
1186     \else
1187       \@l@tempcnta=\line@numR
1188       \edef\rem@inder{\linenumberlist,\number\line@numR,%
1189         \edef\sc@n@list{\def\noexpand\sc@n@list
1190           ###1,\number\@l@tempcnta,###2|{\def\noexpand\rem@inder{###2}}}%
1191       \sc@n@list\expandafter\sc@n@list\rem@inder|
1192       \ifx\rem@inder\empty\advance\@l@tempcnta\@ne\fi
1193     \fi
1194     \ch@ck@l@ckR
1195   \fi
1196 \ifnum\@l@tempcnta=\@l@tempcntb
1197   \if@twocolumn
1198     \if@firstcolumn
1199       \gdef\l@dld@ta{\llap{\leftlinenumR}}}%
1200     \else
1201       \gdef\l@drd@ta{\rlap{\rightlinenumR}}}%
1202     \fi
1203   \else
1204     \@l@tempcntb=\line@marginR
1205     \ifnum\@l@tempcntb>\@ne
1206       \advance\@l@tempcntb by\page@numR
1207     \fi
1208     \ifodd\@l@tempcntb
1209       \gdef\l@drd@ta{\rlap{\rightlinenumR}}}%
1210     \else
1211       \gdef\l@dld@ta{\llap{\leftlinenumR}}}%
1212     \fi

```

```

1213 \fi
1214 \fi
1215 \f@x@l@cksR
1216 \fi
1217 \fi}

```

## 14.5 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 1218
  \leftpstartnumR 1219 \newcommand*{\affixpstart@numL}{%
\rightpstartnumR 1220 \ifsidepstartnum
  \leftpstartnumL 1221 \if@twocolumn
\rightpstartnumL 1222   \if@firstcolumn
  \ifpstartnumR 1223     \gdef\l@dld@ta{\llap{\leftpstartnumL}}}%
1224     \else
1225     \gdef\l@drd@ta{\rlap{\rightpstartnumL}}}%
1226     \fi
1227   \else
1228     \l@dttempcntb=\line@margin
1229     \ifnum\l@dttempcntb>\@ne
1230       \advance\l@dttempcntb \page@num
1231     \fi
1232     \ifodd\l@dttempcntb
1233       \gdef\l@drd@ta{\rlap{\rightpstartnumL}}}%
1234     \else
1235       \gdef\l@dld@ta{\llap{\leftpstartnumL}}}%
1236     \fi
1237   \fi
1238 \fi
1239 }
1240 \newcommand*{\affixpstart@numR}{%
1241 \ifsidepstartnum
1242 \if@twocolumn
1243   \if@firstcolumn
1244     \gdef\l@dld@ta{\llap{\leftpstartnumR}}}%
1245   \else
1246     \gdef\l@drd@ta{\rlap{\rightpstartnumR}}}%
1247   \fi
1248   \else
1249     \l@dttempcntb=\line@marginR
1250     \ifnum\l@dttempcntb>\@ne

```

```

1251     \advance\l@dttempcntb \page@numR
1252     \fi
1253     \ifodd\l@dttempcntb
1254         \gdef\l@drd@ta{\rlap{{\rightpstartnumR}}}%
1255     \else
1256         \gdef\l@dld@ta{\llap{{\leftpstartnumR}}}%
1257     \fi
1258     \fi
1259 \fi
1260 }
1261
1262 \newcommand*{\leftpstartnumL}{
1263 \ifpstartnum
1264 \thepstartL
1265 \kern\linenumsep\global\pstartnumfalse\fi
1266 }
1267 \newcommand*{\rightpstartnumL}{
1268 \ifpstartnum\kern\linenumsep
1269 \thepstartL
1270 \global\pstartnumfalse\fi
1271 }
1272 \newif\ifpstartnumR
1273 \pstartnumRtrue
1274 \newcommand*{\leftpstartnumR}{
1275 \ifpstartnumR
1276 \thepstartR
1277 \kern\linenumsep\global\pstartnumRfalse\fi
1278 }
1279 \newcommand*{\rightpstartnumR}{
1280 \ifpstartnumR\kern\linenumsep
1281 \thepstartR
1282 \global\pstartnumRfalse\fi
1283 }

```

## 14.6 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.

```

1284 \list@create{\inserts@listR}

```

`\add@insertsR` The right text version.

```

\add@inserts@nextR 1285 \newcommand*{\add@insertsR}{%
1286     \global\let\add@inserts@nextR=\relax
1287     \ifx\inserts@listR\empty \else
1288         \ifx\next@insertR\empty
1289             \ifx\insertlines@listR\empty
1290                 \global\noteschanged@true
1291                 \gdef\next@insertR{100000}%
1292             \else

```

```

1293      \gl@p\insertlines@listR\to\next@insertR
1294      \fi
1295  \fi
1296  \ifnum\next@insertR=\absline@numR
1297      \gl@p\inserts@listR\to\@insertR
1298      \@insertR
1299      \global\let\@insertR=\undefined
1300      \global\let\next@insertR=\empty
1301      \global\let\add@inserts@nextR=\add@insertsR
1302  \fi
1303  \fi
1304  \add@inserts@nextR}
1305

```

## 14.7 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.

```

1306 \newcommand*{\add@penaltiesL}{%
1307 \newcommand*{\add@penaltiesR}{%
1308

```

## 14.8 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.

```

1309 \newcommand*{\flush@notesR}{%
1310   \xloop
1311     \ifx\inserts@listR\empty \else
1312       \glp\inserts@listR\to\@insertR
1313       \@insertR
1314       \global\let\@insertR=\undefined
1315   \repeat}
1316

```

## 15 Footnotes

### 15.1 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
1317 \def\printlinesR#1|#2|#3|#4|#5|#6|#7|{\begingroup
1318   \setprintlines{#1}{#2}{#3}{#4}{#5}{#6}%
1319   \ifl@d@pnum #1\fullstop\fi
1320   \ifledplinenum \linenumr@p{#2}\Rlineflag\else \symlinenum\fi
1321   \ifl@d@ssub \fullstop \sublinenumr@p{#3}\fi
1322   \ifl@d@dash \endashchar\fi
1323   \ifl@d@pnum #4\fullstop\fi

```

```

1324 \ifl@d@elin \linenumr@p{#5}\Rlineflag\fi
1325 \ifl@d@esl \ifl@d@elin \fullstop\fi \sublinenumr@p{#6}\fi
1326 \endgroup}
1327
1328 \let\ledsavedprintlines\printlines
1329

```

## 16 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.

```

1330 \list@create{\labelref@listR}
1331

```

`\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.

```

1332 \renewcommand*{\edlabel}[1]{\@bsphack
1333   \ifledRcol
1334     \write\linenum@outR{\string\@lab}%
1335     \ifx\labelref@listR\empty
1336       \xdef\label@refs{\zz@@@}%
1337     \else
1338       \gl@p\labelref@listR\to\label@refs
1339     \fi
1340     \ifvmode
1341       \advancelabel@refs
1342     \fi
1343     \protected@write\@auxout{%
1344       {\string\l@dmake@labelsR\space\thepage|\label@refs|{#1}}%
1345   \else
1346     \write\linenum@out{\string\@lab}%
1347     \ifx\labelref@list\empty
1348       \xdef\label@refs{\zz@@@}%
1349     \else
1350       \gl@p\labelref@list\to\label@refs
1351     \fi
1352     \ifvmode
1353       \advancelabel@refs
1354     \fi
1355     \protected@write\@auxout{%
1356       {\string\l@dmake@labels\space\thepage|\label@refs|{#1}}%
1357   \fi
1358   \@esphack}
1359

```

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

```

1360 \def\l@dmake@labelsR#1|#2|#3|#4{%
1361   \expandafter\ifx\csname the@label#4\endcsname \relax\else
1362     \led@warn@DuplicateLabel{#4}%
1363   \fi
1364   \expandafter\gdef\csname the@label#4\endcsname{#1|#2\Rlineflag|#3}%
1365   \ignorespaces}
1366 \AtBeginDocument{%
1367   \def\l@dmake@labelsR#1|#2|#3|#4{%
1368 }
1369
```

`\@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`, `\@l`, and the `\sub@on` and `\sub@off` commands appearing in the `\linenum@out` file.

```

1370 \renewcommand*{\@lab}{%
1371   \ifledRcol
1372     \xright@appenditem{\linenumr@p{\line@numR}}{|%
1373       \ifsublines@ \sublinenumr@p{\subline@numR}\else 0\fi}%
1374     \to\labelref@listR
1375   \else
1376     \xright@appenditem{\linenumr@p{\line@num}}{|%
1377       \ifsublines@ \sublinenumr@p{\subline@num}\else 0\fi}%
1378     \to\labelref@list
1379   \fi}
1380
```

## 17 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 1381 \newcount\sidenote@marginR
1382 \renewcommand*{\sidenotemargin}[1]{%
1383   \l@dggetsidenote@margin{#1}%
1384   \ifnum\@l@dttempcntb>\m@ne
1385     \ifledRcol
1386       \global\sidenote@marginR=\@l@dttempcntb
1387     \else
1388       \global\sidenote@margin=\@l@dttempcntb
1389     \fi
1390   \fi}}
1391 \sidenotemargin{right}
1392 \global\sidenote@margin=\@ne
1393
```

`\l@dlsnote` The ‘footnotes’ for left, right, and moveable sidenotes. The whole scheme is reminiscent of the critical footnotes code.

```

\l@dcsnote 1394 \renewcommand*{\l@dlsnote}[1]{%
1395   \begingroup%
1396   \newcommand{\content}{#1}%
1397   \ifnumberedpar@
1398     \ifledRcol%
1399       \xright@appenditem{\noexpand\l@dlsnote{\csexpandonce{content}}}%
1400       \to\inserts@listR
1401       \global\advance\insert@countR \@ne%
1402     \else%
1403       \xright@appenditem{\noexpand\l@dlsnote{\csexpandonce{content}}}%
1404       \to\inserts@list
1405       \global\advance\insert@count \@ne%
1406     \fi
1407   \fi\ignorespaces\endgroup}
1408 \renewcommand*{\l@drsnote}[1]{%
1409   \begingroup%
1410   \newcommand{\content}{#1}%
1411   \ifnumberedpar@
1412     \ifledRcol%
1413       \xright@appenditem{\noexpand\l@drsnote{\csexpandonce{content}}}%
1414       \to\inserts@listR
1415       \global\advance\insert@countR \@ne%
1416     \else%
1417       \xright@appenditem{\noexpand\l@drsnote{\csexpandonce{content}}}%
1418       \to\inserts@list
1419       \global\advance\insert@count \@ne%
1420     \fi
1421   \fi\ignorespaces\endgroup}
1422 \renewcommand*{\l@dcsnote}[1]{%
1423   \begingroup%
1424   \newcommand{\content}{#1}%
1425   \ifnumberedpar@
1426     \ifledRcol%
1427       \xright@appenditem{\noexpand\l@dcsnote{\csexpandonce{content}}}%
1428       \to\inserts@listR
1429       \global\advance\insert@countR \@ne%
1430     \else%
1431       \xright@appenditem{\noexpand\l@dcsnote{\csexpandonce{content}}}%
1432       \to\inserts@list
1433       \global\advance\insert@count \@ne%
1434     \fi
1435   \fi\ignorespaces\endgroup}
1436

```

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

```

1437 \newcommand*{\affixside@noteR}{%
1438   \def\sidenotecontent@{}%

```

```

1439 \numdef{\itemcount@}{0}%
1440 \def\do##1{%
1441     \ifnumequal{\itemcount@}{0}%
1442     {%
1443         \appto\sidenotecontent@{##1}}% Not print not separator before the 1st note
1444     {\appto\sidenotecontent@{\sidenotesep ##1}}%
1445     }%
1446     \numdef{\itemcount@}{\itemcount@+1}%
1447 }%
1448 \dolistloop{\l@dcnotesetext}%
1449 \ifnumgreater{\itemcount@}{1}{\eledmac@warning{\itemcount@space sidenotes on line \the\line@numR
1450 \gdef\@templ@d{%
1451 \ifx\@templ@d\l@dcnotesetext \else%
1452     \if@twocolumn%
1453         \if@firstcolumn%
1454             \setl@dlp@rbox{\sidenotecontent@}%
1455         \else%
1456             \setl@drp@rbox{\sidenotecontent@}%
1457         \fi%
1458     \else%
1459         \@l@tempcntb=\sidenote@marginR%
1460         \ifnum\@l@tempcntb>\@ne%
1461             \advance\@l@tempcntb by\page@num%
1462         \fi%
1463         \ifodd\@l@tempcntb%
1464             \setl@drp@rbox{\sidenotecontent@t}%
1465         \else%
1466             \setl@dlp@rbox{\sidenotecontent@}%
1467         \fi%
1468     \fi%
1469 \fi}
1470

```

## 18 Familiar footnotes

`\l@dbfnote` `\l@dbfnote` adds the footnote to the insert list, and `\v1@dbfnote` calls the original `\@footnotetext`.

```

1471 \renewcommand{\l@dbfnote}[1]{%
1472     \ifnumberedpar@
1473     \gdef\@tag{#1}%
1474     \ifledRcol%
1475         \xright@appenditem{\noexpand\v1@dbfnote{\csexpandonce{\@tag}}{\@thefnmark}}%
1476         \to\inserts@listR
1477         \global\advance\insert@countR \@ne%
1478     \else%
1479         \xright@appenditem{\noexpand\v1@dbfnote{\csexpandonce{\@tag}}{\@thefnmark}}%
1480         \to\inserts@list
1481     \global\advance\insert@count \@ne%

```

```

1482   \fi
1483   \fi\ignorespaces}
1484
\normalbfnoteX
1485 \renewcommand{\normalbfnoteX}[2]{%
1486   \ifnumberedpar@
1487   \ifledRcol%
1488   \ifluatex
1489     \footnotelang@lua[R]%
1490   \fi
1491   \ifundefined{xpg@main@language}%if polyglossia
1492 {}%
1493 {\footnotelang@poly[R]}%
1494   \protected@csxdef{thisfootnote}{\csuse{thefootnote#1}}%
1495   \xright@appenditem{\noexpand\vbfnoteX{#1}{#2}{\csexpandonce{thisfootnote}}}%
1496                       \to\inserts@listR
1497   \global\advance\insert@countR \@ne%
1498   \else%
1499   \ifluatex
1500     \footnotelang@lua%
1501   \fi
1502   \ifundefined{xpg@main@language}%if polyglossia
1503 {}%
1504 {\footnotelang@poly}%
1505   \protected@csxdef{thisfootnote}{\csuse{thefootnote#1}}%
1506   \xright@appenditem{\noexpand\vbfnoteX{#1}{#2}{\csexpandonce{thisfootnote}}}%
1507                       \to\inserts@list
1508   \global\advance\insert@count \@ne%
1509   \fi
1510   \fi\ignorespaces}
1511

```

## 19 Verse

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

```

\inserthangingsymbolL
\inserthangingsymbolR 1512 \newif\ifinserthangingsymbolR
1513 \newcommand{\inserthangingsymbolL}{%
1514   \ifinserthangingsymbol%
1515     \ifinstanzaL%
1516       \hangingsymbol%
1517     \fi%
1518   \fi}
1519 \newcommand{\inserthangingsymbolR}{%
1520   \ifinserthangingsymbolR%
1521     \ifinstanzaR%

```

```

1522         \hangingsymbol%
1523     \fi%
1524 \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 1525 \newcommand{\correcthangingL}{%
1526 \ifl@dpadding\else%
1527     \ifinstanzaL%
1528         \ifinserthangingsymbol%
1529             \hskip \@ifundefined{sza@00}{0}{\expandafter%
1530                 \noexpand\csname sza@00\endcsname}\stanzaindentbase%
1531         \fi%
1532     \fi%
1533 \fi}
1534
1535 \newcommand{\correcthangingR}{%
1536 \ifl@dpadding\else%
1537     \ifinstanzaR%
1538         \ifinserthangingsymbolR%
1539             \hskip \@ifundefined{sza@00}{0}{\expandafter%
1540                 \noexpand\csname sza@00\endcsname}\stanzaindentbase%
1541         \fi%
1542     \fi%
1543 \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.

```

1544 \chardef\next=\catcode'\&
1545 \catcode'\&=\active
1546

```

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

```

1547 \newenvironment{astanza}{%
1548     \startstanzahook
1549     \catcode'\&=\active
1550     \global\stanza@count\@ne\stanza@modulo\@ne
1551     \ifnum\usernamecount{sza@00}=\z@
1552         \let\stanza@hang\relax
1553         \let\endlock\relax
1554     \else
1555     %% \interlinepenalty\@M % this screws things up, but I don't know why
1556     \rightskip\z@ plus 1fil\relax

```

```

1557 \fi
1558 \ifnum\usernamecount{szp00@}=\z@
1559   \let\sza@penalty\relax
1560 \fi
1561 \def&{%
1562   \endlock\mbox{}}%
1563   \sza@penalty
1564   \global\advance\stanza@count\@ne
1565   \@astanza@line}%
1566 \def\&{%
1567   \endlock\mbox{}}
1568   \pend
1569   \endstanzaextra}%
1570 \pstart
1571 \@astanza@line
1572 }{ }
1573

```

`\@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.

```

1574 \newcommand*{\@astanza@line}{%
1575   \ifnum\value{stanzaindentrepetition}=0
1576     \parindent=\csname szp@\number\stanza@count
1577       @\endcsname\stanzaindentbase
1578   \else
1579     \parindent=\csname szp@\number\stanza@modulo
1580       @\endcsname\stanzaindentbase
1581     \managestanza@modulo
1582   \fi
1583   \par
1584   \stanza@hang%\mbox{}}%
1585   \ignorespaces}
1586

```

Lastly reset the modified category codes.

```

1587 \catcode'\&=\next
1588

```

## 20 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 1589 \providecommand*{\newnamebox}[1]{%
\unvnamebox 1590   \expandafter\newbox\csname #1\endcsname}
\namebox

```



```

1591 \providecommand*\setnamebox}[1]{%
1592   \expandafter\setbox\csname #1\endcsname}
1593 \providecommand*\unhnamebox}[1]{%
1594   \expandafter\unhbox\csname #1\endcsname}
1595 \providecommand*\unvnamebox}[1]{%
1596   \expandafter\unvbox\csname #1\endcsname}
1597 \providecommand*\namebox}[1]{%
1598   \csname #1\endcsname}
1599

```

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

```

\usenamecount 1600 \providecommand*\newnamecount}[1]{%
1601   \expandafter\newcount\csname #1\endcsname}
1602 \providecommand*\usenamecount}[1]{%
1603   \csname #1\endcsname}
1604

```

## 21 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.

```

1605 \newcount\l@dc@maxchunks
1606 \newcommand*\maxchunks}[1]{\l@dc@maxchunks=#1}
1607   \maxchunks{5120}
1608

```

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

```

\l@dnumpstartsR 1609 \newcount\l@dnumpstartsR
1610

```

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

```

\l@pscR 1611 \newcount\l@dpscL
1612 \newcount\l@dpscR
1613

```

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

```

1614 \newcommand*\l@dsetuprawboxes){%
1615   \@l@dttempcntb=\l@dc@maxchunks
1616   \loop\ifnum\@l@dttempcntb>\z@
1617     \newnamebox{\l@dLcolrawbox\the\@l@dttempcntb}
1618     \newnamebox{\l@dRcolrawbox\the\@l@dttempcntb}
1619     \advance\@l@dttempcntb \m@ne

```

```

1620 \repeat}
1621
\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.
1622 \newcommand*{\l@dssetupmaxlinecounts}{%
1623 \l@dttempcntb=\l@dc@maxchunks
1624 \loop\ifnum\l@dttempcntb>\z@
1625 \newnamecount{\l@dmaxlinesinpar\the\l@dttempcntb}
1626 \advance\l@dttempcntb \m@ne
1627 \repeat}
1628 \newcommand*{\l@dzeromaxlinecounts}{%
1629 \begingroup
1630 \l@dttempcntb=\l@dc@maxchunks
1631 \loop\ifnum\l@dttempcntb>\z@
1632 \global\usenamecount{\l@dmaxlinesinpar\the\l@dttempcntb}=\z@
1633 \advance\l@dttempcntb \m@ne
1634 \repeat
1635 \endgroup}
1636

```

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

```

1637 \AtBeginDocument{%
1638 \l@dssetuppraxboxes
1639 \l@dssetupmaxlinecounts
1640 \l@dzeromaxlinecounts
1641 \l@dnumstartsl=\z@
1642 \l@dnumstartsr=\z@
1643 \l@dpscl=\z@
1644 \l@dpscr=\z@}
1645

```

## 22 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 1646 \newif\ifl@dusedbabel
\l@dusedbabeltrue 1647 \l@dusedbabelfalse

\ifl@dsamelang A flag for checking if the same babel language has been used for both the left and
\l@dsamelangfalse right texts.
\l@dsamelangtrue 1648 \newif\ifl@dsamelang
1649 \l@dsamelangtrue

\l@dchecklang I'm going to use \theledlanguageL and \theledlanguageR to hold the names of
the languages used for the left and right texts. This macro sets \ifl@dsamelang
TRUE if they are the same, otherwise it sets it FALSE.
1650 \newcommand*{\l@dchecklang}{%
1651 \l@dsamelangfalse
1652 \edef\@tempa{\theledlanguageL}\edef\@tempb{\theledlanguageR}%
1653 \ifx\@tempa\@tempb
1654 \l@dsamelangtrue
1655 \fi}
1656

\l@dbbl@set@language In babel the macro \bbl@set@language{<lang>} does the work when the language
<lang> is changed via \selectlanguage. Unfortunately for me, if it is given an
argument in the form of a control sequence it strips off the \ character rather than
expanding the command. I need a version that accepts an argument in the form
\lang without it stripping the \.
1657 \newcommand*{\l@dbbl@set@language}[1]{%
1658 \edef\languageName{#1}%
1659 \select@language{\languageName}%
1660 \if@filesw
1661 \protected@write\auxout{\string\select@language{\languageName}}%
1662 \addtocontents{toc}{\string\select@language{\languageName}}%
1663 \addtocontents{lof}{\string\select@language{\languageName}}%
1664 \addtocontents{lot}{\string\select@language{\languageName}}%
1665 \fi}
1666

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 1667 \providecommand{\selectlanguage}[1]{%
1668 \newcommand*{\l@duselanguage}[1]{%
1669 \gdef\theledlanguageL{
1670 \gdef\theledlanguageR{
1671

```

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

```
1672 \AtBeginDocument{%
1673   \ifundefined{xpg@main@language}{%
1674     \ifundefined{bbl@main@language}{%
```

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

```
1675   \l@dusedbabelfalse
1676   \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.

```
1677   \l@dusedbabeltrue
1678   \let\l@doldselectlanguage\selectlanguage
1679   \let\l@doldbbl@set@language\bbl@set@language
1680   \let\bbl@set@language\l@doldbbl@set@language
1681   \renewcommand{\selectlanguage}[1]{%
1682     \l@doldselectlanguage{#1}%
1683     \ifledRcol \gdef\theledlanguageR{#1}%
1684     \else      \gdef\theledlanguageL{#1}%
1685     \fi}
```

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

```
1686   \renewcommand*{\l@duselanguage}[1]{%
1687     \l@doldselectlanguage{#1}}
```

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

```
1688   \gdef\theledlanguageL{\bbl@main@language}%
1689   \gdef\theledlanguageR{\bbl@main@language}%
1690   }%
1691 }
```

If on `Polyglossia`

```
1692 { \apptocmd{\xpg@set@language}{%
1693   \ifledRcol \gdef\theledlanguageR{#1}%
1694   \else      \gdef\theledlanguageL{#1}%
1695   \fi}%
1696   \let\l@duselanguage\xpg@set@language
1697   \gdef\theledlanguageL{\xpg@main@language}%
1698   \gdef\theledlanguageR{\xpg@main@language}%
1699 % \end{macrocode}
1700 % That's it.
1701 %   \begin{macrocode}
1702 }}
```

## 23 Parallel columns

`\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.

```

1703 \newcommand*{\Columns}{%
1704   \setcounter{pstartL}{\value{pstartLold}}
1705   \setcounter{pstartR}{\value{pstartRold}}
1706   \ifnum\l@dnumstartsL=\l@dnumstartsR\else
1707     \led@err@BadLeftRightPstarts{\the\l@dnumstartsL}{\the\l@dnumstartsR}%
1708   \fi

```

Start a group and zero counters, etc.

```

1709 \begingroup
1710   \l@dzeropenalties
1711   \endgraf\global\num@lines=\prevgraf
1712   \global\num@linesR=\prevgraf
1713   \global\par@line=\z@
1714   \global\par@lineR=\z@
1715   \global\l@dpscL=\z@
1716   \global\l@dpscR=\z@

```

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

```

1717   \check@pstarts
1718   \loop\if@pstarts
1719     \global\pstartnumtrue
1720     \global\pstartnumRtrue

```

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

```

1721     \global\advance\l@dpscL \@ne
1722     \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

```

1723     \checkraw@text
1724     \l@dchecklang
1725 {     \loop\ifaraw@text

```

Grab the next pair of left and right text lines and output them, swapping languages if they differ

```

1726         \ifl@dsamelang
1727         \do@lineL
1728         \do@lineR
1729     \else
1730         \l@duselanguage{\theledlanguageL}%
1731         \do@lineL
1732         \l@duselanguage{\theledlanguageR}%
1733         \do@lineR
1734     \fi
1735     \hb@xt@ \hsize{%
1736         \hfill \unhbox\l@dleftbox
1737         \hfill \columnseparator \hfill
1738         \unhbox\l@drightbox

```

```

1739      }%
1740      \checkdraw@text
1741      \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.

```

1742      \@writelinesinparL
1743      \@writelinesinparR
1744      \check@pstarts
1745          \ifbypstart@
1746              \write\linenum@out{\string\@set[1]}
1747              \resetprevline@
1748          \fi
1749          \ifbypstart@R
1750              \write\linenum@outR{\string\@set[1]}
1751              \resetprevline@
1752          \fi
1753      \addtocounter{pstartL}{1}
1754      \addtocounter{pstartR}{1}
1755      \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.

```

1756      \flush@notes
1757      \flush@notesR
1758      \endgroup
1759      \global\l@dpscL=\z@
1760      \global\l@dpscR=\z@
1761      \global\l@dnumpstartsL=\z@
1762      \global\l@dnumpstartsR=\z@
1763      \ignorespaces
1764      \global\instanzaLfalse
1765      \global\instanzaRfalse}
1766

```

`\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).

```

1767 \newcommand*{\columnseparator}{%
1768   \smash{\rule[-0.2\baselineskip]{\columnrulewidth}{1.05\baselineskip}}}
1769 \newdimen\columnrulewidth
1770 \columnrulewidth=\z@
1771

```

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

```

\@pstartstrue 1772 \newif\if@pstarts
\@pstartsfalse 1773 \newcommand*{\check@pstarts}{%
\check@pstarts

```

```

1774 \@pstartsfalse
1775 \ifnum\l@dnumpstartsL>\l@dpscL
1776   \@pstartstrue
1777 \else
1778   \ifnum\l@dnumpstartsR>\l@dpscR
1779     \@pstartstrue
1780   \fi
1781 \fi
1782 }
1783

```

\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 1784 \newif\ifaraw@text
1785   \araw@textfalse
1786 \newcommand*{\checkraw@text}{%
1787   \araw@textfalse
1788   \ifvbox\namebox{1@dLcolrawbox\the\l@dpscL}
1789     \araw@texttrue
1790   \else
1791     \ifvbox\namebox{1@dRcolrawbox\the\l@dpscR}
1792       \araw@texttrue
1793     \fi
1794   \fi
1795 }
1796

```

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

```

1797 \newcommand*{\@writelinesinparL}{%
1798   \edef\next{%
1799     \write\linenum@out{\string\@pend[\the\@donereallinesL]}}%
1800   \next
1801   \global\@donereallinesL \z@}
1802 \newcommand*{\@writelinesinparR}{%
1803   \edef\next{%
1804     \write\linenum@outR{\string\@pendR[\the\@donereallinesR]}}%
1805   \next
1806   \global\@donereallinesR \z@}
1807

```

## 24 Parallel pages

This is considerably more complicated than parallel columns.

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

```

1808 \newcount\numpagelinesL
1809 \newcount\numpagelinesR
1810 \newcount\l@dminpagelines
1811

```

**\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.

```

1812 \newcommand*{\Pages}{%
1813   \setcounter{pstartL}{\value{pstartLold}}
1814   \setcounter{pstartR}{\value{pstartRold}}
1815   \typeout{}
1816   \typeout{***** PAGES *****}
1817   \ifnum\l@dnumpstartsL=\l@dnumpstartsR\else
1818     \led@err@BadLeftRightPstarts{\the\l@dnumpstartsL}{\the\l@dnumpstartsR}%
1819   \fi

```

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

```

1820 \cleartol@evenpage
1821 \begingroup
1822   \l@dzeropenalties
1823   \endgraf\global\l@num@lines=\prevgraf
1824   \global\l@num@linesR=\prevgraf
1825   \global\l@par@line=\z@
1826   \global\l@par@lineR=\z@
1827   \global\l@dpscL=\z@
1828   \global\l@dpscR=\z@
1829   \writtenlinesLfalse
1830   \writtenlinesRfalse

```

Check if there are chunks to be processed.

```

1831   \check@pstarts
1832   \loop\if@pstarts

```

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

```

1833     \global\advance\l@dpscL \@ne
1834     \global\advance\l@dpscR \@ne

```

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

```

1835     \getlinesfromparlistL
1836     \getlinesfromparlistR
1837     \l@dcalcm@maxoftwo{\cs@linesinparL}{\cs@linesinparR}%
1838     {\usenamecount{\l@dmaxlinesinpar\the\l@dpscL}}%
1839     \check@pstarts
1840   \repeat

```

Zero the counts again, ready for the next bit.

```

1841   \global\l@dpscL=\z@
1842   \global\l@dpscR=\z@

```



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

```
1843 \getlinesfrompagelistL
1844 \getlinesfrompagelistR
1845 \l@dcalc@minoftwo{\@cs@linesonpageL}{\@cs@linesonpageR}%
1846 {\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.

```
1847 \check@pstarts
1848 \if@pstarts
```

Increment the chunk counts to get the first pair.

```
1849 \global\advance\l@dpscL \@ne
1850 \global\advance\l@dpscR \@ne
```

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

```
1851 \global\@donereallinesL=\z@
1852 \global\@donetotallinesL=\z@
1853 \global\@donereallinesR=\z@
1854 \global\@donetotallinesR=\z@
```

Start a loop over the boxes (chunks).

```
1855 \checkraw@text
1856 % \begingroup
1857 { \loop\ifaraw@text
```

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

```
1858 \checkpageL
1859 \l@duselanguage{\theledlanguageL}%
1860 %%% \begingroup
1861 { \loop\ifl@dsamepage
1862
```

Process the next (left) text line, adding it to the page.

```
1863 \do@lineL
1864 \advance\l@numpagelinesL \@ne
1865 \ifshiftedpstarts
1866 \ifdim\ht\l@dleftbox>0pt\hb@xt@ \hsize{\ledstrutL\unhbox\l@dleftbox}\fi%
1867 \else%
1868 \hb@xt@ \hsize{\ledstrutL\unhbox\l@dleftbox}%
1869 \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.

```
1870
1871 \get@nextboxL
1872 \checkpageL
1873 \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.

```

1874      \ifl@dpagfull
1875      \@writelinesonpageL{\the\numpagelinesL}%
1876      \else
1877      \@writelinesonpageL{1000}%
1878      \fi

```

Zero the left page lines count and clear the page to get onto the facing (odd, right) page.

```

1879      \numpagelinesL \z@
1880      \clearl@dleftpage }%

```

Now do the same for the right text.

```

1881      \checkpageR
1882      \l@duselanguage{\theledlanguageR}%
1883 {
1884      \loop\ifl@dsamepage
1885      \do@lineR
1886      \advance\numpagelinesR \@ne
1887      \ifshiftedpstarts
1888      \ifdim\ht\l@drightbox>0pt\hb@xt@ \hsize{\ledstrutR\unhbox\l@drightbox}%
1889      \hb@xt@ \hsize{\ledstrutR\unhbox\l@drightbox}%
1890      \fi
1891      \get@nextboxR
1892      \checkpageR
1893      \repeat
1894      \ifl@dpagfull
1895      \@writelinesonpageR{\the\numpagelinesR}%
1896      \else
1897      \@writelinesonpageR{1000}%
1898      \fi
1899      \numpagelinesR=\z@

```

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

```

1900      \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.

```

1901      \checkraw@text
1902      \ifaraw@text
1903      \getlinesfrompagelistL
1904      \getlinesfrompagelistR
1905      \l@dcalc@minoftwo{\@cs@linesonpageL}{\@cs@linesonpageR}%
1906      {\l@dminpagelines}%
1907      \fi
1908      \repeat}

```

We have now output the text from all the chunks.

```
1909 \fi
```

Make sure that there are no inserts hanging around.

```
1910 \flush@notes
1911 \flush@notesR
1912 \endgroup
```

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

```
1913 \global\l@dpscL=\z@
1914 \global\l@dpscR=\z@
1915 \global\l@dnumpestartsL=\z@
1916 \global\l@dnumpestartsR=\z@
1917 \global\instanzaLfalse
1918 \global\instanzaRfalse
1919 \ignorespaces}
1920
```

`\ledstrutL` Struts inserted into left and right text lines.

```
\ledstrutR 1921 \newcommand*{\ledstrutL}{\strut}
1922 \newcommand*{\ledstrutR}{\strut}
1923
```

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

```
1924 \providecommand{\cleartoevenpage}[1][\@empty]{%
1925 \clearpage
1926 \ifodd\c@page\hbox{ }#1\clearpage\fi}
1927 \newcommand*{\cleartol@devenpage}{%
1928 \ifdim\pagetotal<\topskip% on an empty page
1929 \else
1930 \clearpage
1931 \fi
1932 \ifodd\c@page\hbox{ }\clearpage\fi}
1933 \newcommand*{\clearl@dleftpage}{%
1934 \clearpage
1935 \ifodd\c@page\else
1936 \led@err@LeftOnRightPage
1937 \hbox{ }%
1938 \cleardoublepage
1939 \fi}
1940 \newcommand*{\clearl@drighthouse}{%
1941 \clearpage
1942 \ifodd\c@page
1943 \led@err@RightOnLeftPage
1944 \hbox{ }%
```

```

1945 \cleartoevenpage
1946 \fi}
1947

```

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

```

\@cs@linesinparR 1948 \newcommand*{\getlinesfromparlistL}{%
1949 \ifx\linesinpar@listL\empty
1950 \gdef\@cs@linesinparL{0}%
1951 \else
1952 \gl@p\linesinpar@listL\to\@cs@linesinparL
1953 \fi}
1954 \newcommand*{\getlinesfromparlistR}{%
1955 \ifx\linesinpar@listR\empty
1956 \gdef\@cs@linesinparR{0}%
1957 \else
1958 \gl@p\linesinpar@listR\to\@cs@linesinparR
1959 \fi}
1960

```

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

```

\@cs@linesonpageR 1961 \newcommand*{\getlinesfrompagelistL}{%
1962 \ifx\linesonpage@listL\empty
1963 \gdef\@cs@linesonpageL{1000}%
1964 \else
1965 \gl@p\linesonpage@listL\to\@cs@linesonpageL
1966 \fi}
1967 \newcommand*{\getlinesfrompagelistR}{%
1968 \ifx\linesonpage@listR\empty
1969 \gdef\@cs@linesonpageR{1000}%
1970 \else
1971 \gl@p\linesonpage@listR\to\@cs@linesonpageR
1972 \fi}
1973

```

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

```

1974 \newcommand*{\@writelinesonpageL}[1]{%
1975 \edef\next{\write\linenum@out{\string\@lopL{#1}}}%
1976 \next}
1977 \newcommand*{\@writelinesonpageR}[1]{%
1978 \edef\next{\write\linenum@outR{\string\@lopR{#1}}}%
1979 \next}
1980

```

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

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

```

1981 \newcommand*{\l@dcalc@maxoftwo}[3]{%
1982   \ifnum #2>#1\relax
1983     #3=#2\relax
1984   \else
1985     #3=#1\relax
1986   \fi}
1987 \newcommand*{\l@dcalc@minoftwo}[3]{%
1988   \ifnum #2<#1\relax
1989     #3=#2\relax
1990   \else
1991     #3=#1\relax
1992   \fi}
1993

```

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

```

\checkpageL 1994 \newif\ifl@dsamepage
\checkpageR 1995   \l@dsamepagetrue
1996 \newif\ifl@dpagetrue
1997 \newcommand*{\checkpageL}{%
1998   \l@dpagetrue
1999   \l@dsamepagetrue
2000   \check@goal
2001   \ifdim\pagetotal<\ledthegoal
2002     \ifnum\umpagelinesL<\l@dmipagelines
2003       \else
2004         \l@dsamepagefalse
2005         \l@dpagetruefalse
2006       \fi
2007     \else
2008       \l@dsamepagefalse
2009       \l@dpagetrue
2010     \fi}
2011 \newcommand*{\checkpageR}{%
2012   \l@dpagetrue
2013   \l@dsamepagetrue
2014   \check@goal
2015   \ifdim\pagetotal<\ledthegoal
2016     \ifnum\umpagelinesR<\l@dmipagelines
2017       \else
2018         \l@dsamepagefalse
2019         \l@dpagetruefalse
2020       \fi

```

```

2021 \else
2022     \l@dsamepagefalse
2023     \l@dpagfulltrue
2024 \fi}
2025

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

2026 \newdimen\ledthegoal
2027 \ifshiftedpstarts
2028     \newcommand*{\goalfraction}{0.95}
2029 \else
2030     \newcommand*{\goalfraction}{0.9}
2031 \fi
2032
2033 \newcommand*{\check@goal}{%
2034     \ledthegoal=\goalfraction\pagegoal}
2035

\ifwrittenlinesL Booleans for whether line data has been written to the section file.
\ifwrittenlinesL 2036 \newif\ifwrittenlinesL
2037 \newif\ifwrittenlinesR
2038

\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.

2039 \newcommand*{\get@nextboxL}{%
2040     \ifvbox\namebox{\l@dLcolrawbox\the\l@dpscL}% box is not empty

        The current box is not empty; do nothing.

2041     \else%                                box is empty

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

2042         \ifnum\usenamecount{\l@dmaxlinesinpar\the\l@dpscL}>\@donetotallinesL
2043         \else

            Sufficient lines have been output.

2044             \ifwrittenlinesL
2045             \else

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

2046                 \@writelinesinparL
2047                 \writtenlinesLtrue
2048             \fi
2049             \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.

```

```

2050     \writtenlinesLfalse
2051     \ifbypstart@
2052         \ifnum\value{pstartL}<\value{pstartLold}
2053         \else
2054             \global\line@num=0
2055             \resetprevline@
2056         \fi
2057     \fi
2058     \addtocounter{pstartL}{1}
2059     \global\pstartnumtrue
2060     \l@dcalc@maxoftwo{\the\usernamecount{1@dmaxlinesinpar\the\l@dpscL}}%
2061                     {\the\@donetotallinesL}%
2062                     {\usernamecount{1@dmaxlinesinpar\the\l@dpscL}}%
2063     \global\@donetotallinesL \z@
2064     \global\advance\l@dpscL \@ne
2065 \fi
2066 \fi
2067 \fi}

2068 \newcommand*{\get@nextboxR}{%
2069     \ifvbox\namebox{1@dRcolrawbox\the\l@dpscR}% box is not empty
2070     \else% box is empty
2071         \ifnum\usernamecount{1@dmaxlinesinpar\the\l@dpscR}>\@donetotallinesR
2072         \else
2073             \ifwrittenlinesR
2074             \else
2075                 \@writelinesinparR
2076                 \writtenlinesRtrue
2077             \fi
2078             \ifnum\l@dnumpstartsR>\l@dpscR
2079                 \writtenlinesRfalse
2080                 \ifbypstart@R
2081                     \ifnum\value{pstartR}<\value{pstartRold}
2082                     \else
2083                         \global\line@numR=0
2084                         \resetprevline@
2085                     \fi
2086                 \fi
2087                 \addtocounter{pstartR}{1}
2088                 \global\pstartnumRtrue
2089                 \l@dcalc@maxoftwo{\the\usernamecount{1@dmaxlinesinpar\the\l@dpscR}}%
2090                                 {\the\@donetotallinesR}%
2091                                 {\usernamecount{1@dmaxlinesinpar\the\l@dpscR}}%
2092                 \global\@donetotallinesR \z@
2093                 \global\advance\l@dpscR \@ne
2094             \fi
2095         \fi
2096     \fi}
2097

```

**25 The End**

i/code<sub>i</sub>



## Appendix A Some things to do when changing version

### Appendix A.1 Migration to eledpar 1.4.3

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

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

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

See the two following examples:

With standard `\hangingsymbol`:

A very long verse should be sometime hanged. The position of the hang 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>\@M</code> . . . . .	1555
<code>\&amp;</code> . . . . .	1544, 1545, 1549, 1566, 1587	<code>\@adv</code> . . . . . <u>352</u> , 631, 632

- \@afterindentfalse ..... 754
  - \@arabic ..... 209, 210, 803, 806
  - \@astanza@line ..... 1565, 1571, 1574
  - \@auxout ..... 1343, 1355, 1661
  - \@chapter ..... 755
  - \@cs@linesinparL ..... 1837, 1948
  - \@cs@linesinparR ..... 1837, 1948
  - \@cs@linesonpageL .. 1845, 1905, 1961
  - \@cs@linesonpageR .. 1845, 1905, 1961
  - \@currentlabel ..... 846, 886
  - \@donereallinesL .....
    - ..... 930, 959, 1799, 1801, 1851
  - \@donereallinesR .....
    - ..... 930, 996, 1804, 1806, 1853
  - \@donetotallinesL ..... 930,
    - 960, 963, 1852, 2042, 2061, 2063
  - \@donetotallinesR ..... 930,
    - 997, 1000, 1854, 2071, 2090, 2092
  - \@insertR ..... 1297–1299, 1312–1314
  - \@l ..... 275, 600
  - \@l@dttempcnta ..... 425,
    - 427, 429, 430, 434, 436, 438,
    - 439, 1050, 1089, 1090, 1092,
    - 1094, 1097, 1098, 1115–1119,
    - 1121, 1128, 1133, 1137, 1145,
    - 1150, 1154, 1187, 1190, 1192, 1196
  - \@l@dttempcntb . 168, 170, 172, 1080,
    - 1081, 1128, 1133, 1137, 1145,
    - 1150, 1154, 1179, 1183, 1196,
    - 1204–1206, 1208, 1228–1230,
    - 1232, 1249–1251, 1253, 1384,
    - 1386, 1388, 1459–1461, 1463,
    - 1615–1619, 1623–1626, 1630–1633
  - \@l@reg ..... 324
  - \@l@regR ..... 275
  - \@lab ..... 544, 1334, 1346, 1370
  - \@lock ..... 946, 1031
  - \@lockR .... 60, 297, 299, 301, 314,
    - 459, 475, 476, 478, 479, 507,
    - 508, 510, 983, 1014, 1056, 1058,
    - 1059, 1061, 1142, 1159, 1161, 1163
  - \@lopL ..... 568, 1975
  - \@lopR ..... 568, 1978
  - \@nobreakfalse ..... 812, 853
  - \@nobreaktrue ..... 810, 814, 851, 855
  - \@oldnobreak 810, 812, 851, 853, 900, 916
  - \@pend ..... 559, 1799
  - \@pendR ..... 559, 1804
  - \@pstartsfalse ..... 1772
  - \@pstartstrue ..... 1772
  - \@ref ..... 531, 604, 608
  - \@ref@reg ..... 557
  - \@schapter ..... 755
  - \@set ..... 384, 638, 639, 1746, 1750
  - \@tag ..... 668, 691, 1473
  - \@temp ..... 1652
  - \@templ@d ..... 1450, 1451
  - \@writelinesinparL .. 1742, 1797, 2046
  - \@writelinesinparR .. 1743, 1797, 2075
  - \@writelinesonpageL . 1875, 1877, 1974
  - \@writelinesonpageR . 1895, 1897, 1974
  - \@xloop ..... 1310
- A**
- \absline@num ..... 418, 432, 451, 1022
  - \absline@numR 58, 226, 277, 280, 283,
    - 415, 423, 444, 463, 497, 525,
    - 536, 1005, 1042, 1043, 1080, 1296
  - \actionlines@list .....
    - ..... 267, 270, 418, 432, 451
  - \actionlines@listR .....
    - ..... 230, 245, 259, 262, 415,
    - 423, 444, 463, 497, 525, 1102, 1105
  - \actions@list . 271, 419, 439, 453, 455
  - \actions@listR .....
    - . 230, 246, 263, 416, 430, 446,
    - 448, 465, 474, 499, 506, 526, 1106
  - \add@inserts ..... 952
  - \add@inserts@nextR ..... 1285
  - \add@insertsR ..... 989, 1285
  - \add@penaltiesL ..... 958, 1306
  - \add@penaltiesR ..... 995, 1306
  - \addtocontents ..... 1662–1664
  - \addtocounter .....
    - . 902, 918, 1753, 1754, 2058, 2087
  - \advancelabel@refs ..... 1341, 1353
  - \advanceline ..... 630, 661
  - \affixline@num ..... 950
  - \affixline@numR ..... 987, 1112
  - \affixpstart@numL ..... 949, 1218
  - \affixpstart@numR ..... 986, 1218
  - \affixside@note ..... 953
  - \affixside@noteR ..... 990, 1437
  - \appto ..... 1443, 1444
  - \apptocmd ..... 1692
  - \araw@textfalse ..... 1784
  - \araw@texttrue ..... 1784
  - astanza (environment) ..... 8, 1547
  - \AtBeginDocument ... 1366, 1637, 1672

**B**

`\ballast@count` ..... 1040, 1045  
`\bbl@main@language` ..... 1688, 1689  
`\bbl@set@language` ..... 1679, 1680  
`\beginnumbering` .. 6, 36, 761, 779, 817  
`\beginnumberingR` ... 49, 117, 779, 858  
`\bfseries` ..... 803, 806  
`\bypage@Rfalse` ..... 137, 152, 157  
`\bypage@Rtrue` ..... 137, 147  
`\bypstart@Rfalse` ..... 137, 148, 158  
`\bypstart@Rtrue` ..... 137, 153

**C**

`\c@ballast` ..... 1045  
`\c@chapter` ..... 100  
`\c@chapterR` ..... 100  
`\c@firstlinenumR` ..... 177, 1185  
`\c@firstsublinenumR` ..... 181, 1180  
`\c@linenumincrementR` ..... 177, 1185  
`\c@page` ... 600, 1926, 1932, 1935, 1942  
`\c@pstartL` ..... 803  
`\c@pstartR` ..... 806  
`\c@section` ..... 101  
`\c@sectionR` ..... 101  
`\c@sublinenumincrementR` .. 181, 1180  
`\c@subsection` ..... 102  
`\c@subsectionR` ..... 102  
`\c@subsubsection` ..... 103  
`\c@subsubsectionR` ..... 103  
`\ch@ck@l@ckR` ..... 1112  
`\ch@cksub@l@ckR` ..... 1112  
`\ch@cksub@lockR` ..... 1181  
`\chapter` ..... 741, 742, 750  
`\chapterinpages` ..... 734, 742, 752  
`\chardef` ..... 1544  
`\check@goal` ..... 2000, 2014, 2026  
`\check@pstarts` .....  
     1717, 1744, 1772, 1831, 1839, 1847  
`\checkpageL` ..... 1858, 1872, 1994  
`\checkpageR` ..... 1881, 1892, 1994  
`\checkraw@text` .....  
     .... 1723, 1740, 1784, 1855, 1901  
`\cleardoublepage` ..... 1938  
`\clearl@dleftpage` ..... 1880, 1924  
`\clearl@drightpage` ..... 1900, 1924  
`\cleartoevenpage` ..... 1924  
`\cleartol@devenpage` ..... 1820, 1924  
`\closeout` ..... 581, 587, 590, 594  
`\columnrulewidth` ..... 4, 1767  
`\Columns` ..... 3, 1703

`\columnseparator` ..... 4, 1737, 1767  
`\content` ..... 1396, 1410, 1424  
`\correcthangingL` ..... 955, 1525  
`\correcthangingR` ..... 992, 1525  
`\countLline` ..... 925, 936  
`\countRline` ..... 925, 973  
`\critext` ..... 666  
`\csexpandonce` 1399, 1403, 1413, 1417,  
     1427, 1431, 1475, 1479, 1495, 1506  
`\csuse` ..... 1494, 1505

**D**

`\DeclareOption` ..... 8, 9  
`\def@tempb` ..... 155  
`\dimen` ..... 617, 618, 622–624, 628  
`\divide` ..... 1117  
`\do@actions` ..... 1023  
`\do@actions@fixedcodeR` ..... 1049  
`\do@actions@nextR` ..... 1049  
`\do@actionsR` ..... 1006, 1049  
`\do@ballast` ..... 1024  
`\do@ballastR` ..... 1007, 1040  
`\do@insidelineLhook` ..... 955, 967  
`\do@insidelineRhook` ..... 967, 992  
`\do@lineL` ..... 935, 1727, 1731, 1863  
`\do@lineLhook` ..... 940, 967  
`\do@lineR` ..... 972, 1728, 1733, 1884  
`\do@lineRhook` ..... 967, 977  
`\do@lockoff` ..... 494  
`\do@lockoffL` ..... 518  
`\do@lockoffR` ..... 494  
`\do@lockon` ..... 459  
`\do@lockonL` ..... 491  
`\do@lockonR` ..... 459  
`\dolistloop` ..... 1448  
`\dummy@ref` ..... 540

**E**

`\edfont@info` ..... 717, 720, 726, 729  
`\edlabel` ..... 1332  
`\edtext` ..... 689  
`\eledmac@error` ..... 21, 24, 27, 29  
`\eledmac@warning` ..... 1449  
`\empty` .. 81, 84, 259, 267, 679, 702,  
     715, 724, 826, 867, 1102, 1184,  
     1192, 1287–1289, 1300, 1311,  
     1335, 1347, 1949, 1955, 1962, 1968  
`\end@lemmas` ..... 679, 680, 702, 703  
`\endashchar` ..... 1322  
`\endgraf` ..... 896, 912, 1711, 1823

`\endline@num` ..... 547, 553  
`\endlock` ..... 650, 1553, 1562, 1567  
`\endnumbering` ..... 6, 39, 74, 121, 780  
`\endnumberingR` 52, 74, 106, 116, 129, 780  
`\endpage@num` ..... 546, 553  
`\endstanzaextra` ..... 1569  
`\endsub` ..... 617  
`\endsubline@num` ..... 548, 554  
environments:  
    `astanza` ..... 8, 1547  
    `Leftside` ..... 5, 759  
    `pages` ..... 4, 734  
    `pairs` ..... 3, 734  
    `Rightside` ..... 5, 777  
`\extensionchars` . 47, 66, 112, 126, 134

## F

`\f@x@l@cksR` ..... 1112  
`\first@linenum@out@Rfalse` .. 576, 582  
`\first@linenum@out@Rtrue` ..... 576  
`\firstlinenum` ..... 5, 186  
`\firstsublinenum` ..... 5, 186  
`\fix@page` ..... 320, 327  
`\flag@end` .....  
    . 601, 675, 685, 686, 698, 708, 709  
`\flag@start` .....  
    . 601, 674, 675, 686, 697, 698, 709  
`\flush@notes` ..... 1756, 1910  
`\flush@notesR` ..... 1309, 1757, 1911  
`\footnotelang@lua` ..... 1489, 1500  
`\footnotelang@poly` ..... 1493, 1504  
`\fullstop` . 222, 1319, 1321, 1323, 1325

## G

`\get@linelistfile` ..... 255  
`\get@nextboxL` ..... 1871, 2039  
`\get@nextboxR` ..... 1891, 2039  
`\getline@numL` ..... 945, 1021  
`\getline@numR` ..... 982, 1004  
`\getlinesfrompagelistL` .....  
    ..... 1843, 1903, 1961  
`\getlinesfrompagelistR` .....  
    ..... 1844, 1904, 1961  
`\getlinesfromparlistL` ... 1835, 1948  
`\getlinesfromparlistR` ... 1836, 1948  
`\gl@p` ..... 262, 263,  
    270, 271, 680, 703, 719, 728,  
    1105, 1106, 1293, 1297, 1312,  
    1338, 1350, 1952, 1958, 1965, 1971  
`\goalfraction` ..... 4, 2026

## H

`\hangingsymbol` ..... 9, 1516, 1522  
`\hb@xt@` ... 948, 955, 962, 985, 992,  
    999, 1735, 1866, 1868, 1887, 1889  
`\hsize` ..... 844,  
    884, 1735, 1866, 1868, 1887, 1889

## I

`\if@filesw` ..... 1660  
`\if@firstcolumn` 1198, 1222, 1243, 1453  
`\if@ledgroup` ..... 586  
`\if@nobreak` ..... 809, 850  
`\if@pstarts` .... 1718, 1772, 1832, 1848  
`\if@RTL` ..... 675, 686, 698, 709  
`\ifaraw@text` ... 1725, 1784, 1857, 1902  
`\ifautopar` ..... 836, 877  
`\ifbypage@` ..... 343  
`\ifbypage@R` ..... 137, 333, 1084  
`\ifbypstart@` ..... 561, 1745, 2051  
`\ifbypstart@R` ... 137, 565, 1749, 2080  
`\ifdim` ..... 618, 622, 624,  
    628, 1866, 1887, 1928, 2001, 2015  
`\iffirst@linenum@out@R` .... 576, 580  
`\ifinserthangingsymbol` .. 1514, 1528  
`\ifinserthangingsymbolR` .....  
    ..... 1512, 1520, 1538  
`\ifinstanzaL` .... 757, 757, 1515, 1527  
`\ifinstanzaR` .... 757, 758, 1521, 1537  
`\ifl@d@dash` ..... 1322  
`\ifl@d@elin` ..... 1324, 1325  
`\ifl@d@esl` ..... 1325  
`\ifl@d@pnum` ..... 1319, 1323  
`\ifl@d@ssub` ..... 1321  
`\ifl@dpagefull` ..... 1874, 1894, 1994  
`\ifl@dpadding` ..... 11, 1526, 1536  
`\ifl@dpairing` ..... 11, 78  
`\ifl@dsamelang` ..... 1648, 1726  
`\ifl@dsamepage` .... 1861, 1883, 1994  
`\ifl@dskipnumber` ..... 1175  
`\ifl@dusedbabel` ..... 1646  
`\iflabelpstart` ..... 846, 886  
`\ifledplinenum` ..... 1320  
`\ifledRcol` ..... 11, 169, 191,  
    195, 199, 203, 242, 257, 321,  
    330, 354, 368, 385, 402, 414,  
    422, 443, 488, 515, 524, 533,  
    602, 612, 619, 625, 631, 638,  
    646, 651, 655, 660, 670, 693,  
    714, 1333, 1371, 1385, 1398,  
    1412, 1426, 1474, 1487, 1683, 1693

- \ifluatex ..... 1488, 1499
  - \ifnoteschanged@ ..... 88
  - \ifnumberedpar@ ... 819, 860, 892,  
908, 1397, 1411, 1425, 1472, 1486
  - \ifnumbering ..... 37, 142, 815, 889
  - \ifnumberingR ... 50, 75, 108, 856, 905
  - \ifnumberline ..... 1008, 1025, 1174
  - \ifnumberpstart ... 837, 878, 901, 917
  - \ifnumequal ..... 1441
  - \ifnumgreater ..... 1449
  - \ifodd ..... 1208, 1232,  
1253, 1463, 1926, 1932, 1935, 1942
  - \ifpst@rtedL ..... 32, 823
  - \ifpst@rtedR ..... 32, 864
  - \ifpstartnum ..... 1263, 1268
  - \ifpstartnumR ..... 1218
  - \ifshiftedpstarts 5, 1865, 1886, 2027
  - \ifsidepstartnum 838, 879, 1220, 1241
  - \ifsublines@ ..... 220,  
309, 353, 386, 393, 424, 433,  
445, 452, 464, 498, 552, 554,  
1009, 1026, 1091, 1178, 1373, 1377
  - \ifvbox 937, 974, 1788, 1791, 2040, 2069
  - \ifvmode ..... 1340, 1352
  - \ifwrittenlinesL ..... 2036, 2044
  - \ifwrittenlinesR ..... 2037, 2073
  - \initnumbering@reg ..... 45
  - \initnumbering@sectcmd ..... 70
  - \initnumbering@sectcountR ... 71, 95
  - \insert@count ..... 530, 608, 671,  
694, 1405, 1419, 1433, 1481, 1508
  - \insert@countR ... 531, 604, 670,  
693, 1401, 1415, 1429, 1477, 1497
  - \inserthangingsymbolfalse ..... 946
  - \inserthangingsymbolL ... 955, 1512
  - \inserthangingsymbolR ... 992, 1512
  - \inserthangingsymbolRfalse ..... 983
  - \inserthangingsymbolRtrue ..... 983
  - \inserthangingsymboltrue ..... 946
  - \insertlines@listR .....  
... 81, 230, 244, 536, 1289, 1293
  - \inserts@list .....  
825, 1404, 1418, 1432, 1480, 1507
  - \inserts@listR .....  
... 866, 1284, 1287, 1297, 1311,  
1312, 1400, 1414, 1428, 1476, 1496
  - \instanzaLfalse ..... 1764, 1917
  - \instanzaLtrue ..... 768
  - \instanzaRfalse ..... 1765, 1918
  - \instanzaRtrue ..... 790
  - \interlinepenalty ..... 1555
  - \itemcount@ .... 1439, 1441, 1446, 1449
- L**
- \l@d@nums ..... 717, 720, 726, 729
  - \l@d@set ..... 401, 646, 647
  - \l@dbbl@set@language .... 1657, 1680
  - \l@dbfnote ..... 1471
  - \l@dc@maxchunks ..... 831, 833,  
872, 874, 1605, 1615, 1623, 1630
  - \l@dcalc@maxoftwo .....  
... 1837, 1981, 2060, 2089
  - \l@dcalc@minoftwo .. 1845, 1905, 1981
  - \l@dcalcnun ..... 1112
  - \l@dchecklang ..... 1650, 1724
  - \l@dchset@num ..... 276, 279, 401
  - \l@dcsnote ..... 1394
  - \l@dcsnotetext ..... 1448, 1451
  - \l@emptyd@ta ..... 941, 978
  - \l@dend@stuff ... 48, 67, 113, 127, 135
  - \l@dgetline@margin ..... 167
  - \l@dgetsidenote@margin ..... 1383
  - \l@dld@ta ..... 951, 988,  
1199, 1211, 1223, 1235, 1244, 1256
  - \l@dleftbox .....  
... 922, 947, 962, 1736, 1866, 1868
  - \l@dlinenumR ..... 212
  - \l@dlsn@te ..... 954, 991
  - \l@dlsnote ..... 1394
  - \l@dmake@labels ..... 1356
  - \l@dmake@labelsR ..... 1344, 1360
  - \l@dminpagelines .....  
... 1808, 1846, 1906, 2002, 2016
  - \l@dnumpstartsL . 41, 830, 831, 833,  
835, 1609, 1641, 1706, 1707,  
1761, 1775, 1817, 1818, 1915, 2049
  - \l@dnumpstartsR . 54, 871, 872, 874,  
876, 1609, 1642, 1706, 1707,  
1762, 1778, 1817, 1818, 1916, 2078
  - \l@doldbbl@set@language ..... 1679
  - \l@doldselectlanguage 1678, 1682, 1687
  - \l@dpagefullfalse ..... 1994
  - \l@dpagefulltrue ..... 1994
  - \l@dpagingfalse ..... 13, 736, 749
  - \l@dpagingtrue ..... 744
  - \l@dpairingfalse ..... 11, 738, 748
  - \l@dpairingtrue ..... 735, 743
  - \l@dpscL 937, 942, 1611, 1643, 1715,  
1721, 1759, 1775, 1788, 1827,

- 1833, 1838, 1841, 1849, 1913,
- 2040, 2042, 2049, 2060, 2062, 2064
- \l@dpscR . . . . 974, 979, 1612, 1644,
- 1716, 1722, 1760, 1778, 1791,
- 1828, 1834, 1842, 1850, 1914,
- 2069, 2071, 2078, 2089, 2091, 2093
- \l@drd@eta . . . . . 955, 992,
- 1201, 1209, 1225, 1233, 1246, 1254
- \l@drightbox . . . . .
- . . . 922, 984, 999, 1738, 1887, 1889
- \l@drsn@te . . . . . 956, 993
- \l@drsnote . . . . . 1394
- \l@dsamelangfalse . . . . . 1648, 1651
- \l@dsamelangtrue . . . . . 1648, 1654
- \l@dsamepagfalse . . . . . 1994
- \l@dsamepagtrue . . . . . 1994
- \l@dsetupmaxlinecounts . . 1622, 1639
- \l@dsetuprawboxes . . . . . 1614, 1638
- \l@dskipnumberfalse . . . . . 1176
- \l@dskipnumbertrue . . . . . 1072
- \l@dunhbox@line . . . . . 955, 992
- \l@dusedbabelfalse . . . . . 1646, 1675
- \l@dusedbabeltrue . . . . . 1646, 1677
- \l@duselanguage . . . . .
- . . . . 1667, 1730, 1732, 1859, 1882
- \l@dzeromaxlinecounts . . . 1622, 1640
- \l@dzeropenalties 895, 911, 1710, 1822
- \l@pscl . . . . . 1611
- \l@pscR . . . . . 1611
- \label@refs . . . . .
- 1336, 1338, 1344, 1348, 1350, 1356
- \labelref@list . . . . . 1347, 1350, 1378
- \labelref@listR 1330, 1335, 1338, 1374
- \languagename . . 1658, 1659, 1661–1664
- \last@page@num . . . . . 341, 347
- \last@page@numR . . . . . 327
- \lastbox . . . . . 944, 981
- \lastskip . . . . . 617, 623
- \Lcolwidth . . . 4, 15, 745, 844, 948, 962
- \led@err@BadLeftRightPstarts . . .
- . . . . . 23, 1707, 1818
- \led@err@LeftOnRightPage . . . 26, 1936
- \led@err@LineationInNumbered . . . 143
- \led@err@NumberingNotStarted . . . 92
- \led@err@NumberingShouldHaveStarted
- . . . . . 115
- \led@err@NumberingStarted . . . . 38, 51
- \led@err@PendNoPstart . . . . . 893, 909
- \led@err@PendNotNumbered . . . 890, 906
- \led@err@PstartInPstart . . . 820, 861
- \led@err@PstartNotNumbered . 816, 857
- \led@err@RightOnLeftPage . . . 26, 1943
- \led@err@TooManyPstarts 20, 832, 873
- \led@mess@NotesChanged . . . . . 89
- \led@mess@SectionContinued . . . . .
- . . . . . 111, 125, 133
- \led@warn@BadAction . . . . . 1074
- \led@warn@BadAdvancelineLine 371, 377
- \led@warn@BadAdvancelineSubline .
- . . . . . 357, 363
- \led@warn@BadLineation . . . . . 160
- \led@warn@BadSetline . . . . . 636
- \led@warn@BadSetlinenum . . . . . 644
- \led@warn@DuplicateLabel . . . . . 1362
- \ledllfill . . . . . 955, 992
- \ledRcolfalse . . . . . 14, 760, 792
- \ledRcoltrue . . . . . 778
- \ledrlfill . . . . . 955, 992
- \ledsavedprintlines . . . . . 7, 1317
- \ledstrutL . . . . . 1866, 1868, 1921
- \ledstrutR . . . . . 1887, 1889, 1921
- \ledthegoal . . . . . 2001, 2015, 2026
- \leftlinenumR . . . . . 212, 1199, 1211
- \leftpstartnumL . . . . . 1218
- \leftpstartnumR . . . . . 1218
- Leftside (environment) . . . . . 5, 759
- \Leftsidehook . . . . . 766, 772
- \Leftsidehookend . . . . . 771, 772
- \line@list . . . . . 724, 728
- \line@list@stuff . . . . . 47, 126
- \line@list@stuffR . . 66, 112, 134, 578
- \line@listR . 84, 230, 243, 554, 715, 719
- \line@margin . . . . . 172, 1228
- \line@marginR . . . . . 165, 1204, 1249
- \line@num . 344, 375, 376, 378, 396,
- 407, 408, 436, 561, 1032, 1376, 2054
- \line@numR . . . . . 59, 219, 226,
- 281, 315, 334, 369, 370, 372,
- 389, 403, 404, 427, 547, 551,
- 565, 1015, 1085, 1094, 1183,
- 1185, 1187, 1188, 1372, 1449, 2083
- \lineation . . . . . 787
- \lineationR . . . . . 141, 787
- \linenum@out . . . . . 607,
- 615, 620, 626, 632, 639, 647,
- 652, 656, 1346, 1746, 1799, 1975
- \linenum@outR . . . . .
- . 575, 581, 583, 587, 588, 590,
- 591, 594, 595, 600, 603, 613,

- 619, 625, 631, 638, 646, 651,  
655, 660, 1334, 1750, 1804, 1978  
`\linenumberlist` ..... 1184, 1188  
`\linenumincrement` ..... 5, 186  
`\linenummargin` ..... 165  
`\linenumr@p` .... 1320, 1324, 1372, 1376  
`\linenumrepR` ..... 209, 219  
`\linenumsep` .....  
     . 214, 216, 1265, 1268, 1277, 1280  
`\linesinpar@listL` .....  
     ..... 235, 251, 562, 1949, 1952  
`\linesinpar@listR` .....  
     ..... 235, 247, 566, 1955, 1958  
`\linesonpage@listL` 252, 570, 1962, 1965  
`\linesonpage@listR` 248, 573, 1968, 1971  
`\list@clear` .....  
     . 243–248, 251, 252, 254, 825, 866  
`\list@clearing@reg` ..... 250  
`\list@create` .....  
     ... 230–233, 235–237, 1284, 1330  
`\lock@disp` ..... 1144, 1148, 1153  
`\lock@off` .... 485, 486, 494, 655, 656  
`\lock@on` ..... 651, 652
- M**
- `\managestanza@modulo` ..... 1581  
`\maxchunks` ..... 3, 1605  
`\maxlinesinpar@list` ..... 235, 254  
`\memorydump` ..... 6, 765, 783  
`\memorydumpL` ..... 120, 765  
`\memorydumpR` ..... 120, 783  
`\message` ..... 46, 65  
`\multiply` ..... 1118
- N**
- `\n@num` ..... 522, 660  
`\n@num@reg` ..... 528  
`\namebox` ..... 937, 942, 974,  
     979, 1589, 1788, 1791, 2040, 2069  
`\NeedsTeXFormat` ..... 2  
`\new@line` ..... 955  
`\new@lineR` ..... 599, 992  
`\newbox` ..... 798, 922, 923, 1590  
`\newcounter` ..... 95–98, 177,  
     179, 181, 183, 801, 802, 804, 805  
`\newif` . 5, 12, 33, 137, 138, 576, 757,  
     758, 1272, 1512, 1646, 1648,  
     1772, 1784, 1994, 1996, 2036, 2037  
`\newnamebox` ..... 1589, 1617, 1618  
`\newnamecount` ..... 1600, 1625
- `\newwrite` ..... 575  
`\next@action` ..... 271  
`\next@actionline` ..... 268, 270  
`\next@actionlineR` .....  
     . 260, 262, 1043, 1081, 1103, 1105  
`\next@actionR` ..... 263, 1044,  
     1082, 1083, 1088, 1089, 1097, 1106  
`\next@insert` ..... 826  
`\next@insertR` .....  
     867, 1288, 1291, 1293, 1296, 1300  
`\next@page@num` ..... 348, 419  
`\next@page@numR` 63, 284, 286, 338, 416  
`\no@expands` ..... 668, 691  
`\normal@pars` ..... 77, 829, 870  
`\normalbfnoteX` ..... 1485  
`\noteschanged@true` .....  
     ..... 82, 85, 716, 725, 1290  
`\num@lines` ..... 896, 1711, 1823  
`\num@linesR` .... 797, 912, 1712, 1824  
`\numberedpar@true` ..... 845, 885  
`\numberingRfalse` ..... 76  
`\numberingRtrue` ..... 56, 106, 130  
`\numberingtrue` ..... 43, 122  
`\numberpstartfalse` ..... 7  
`\numberpstarttrue` ..... 7  
`\numdef` ..... 1439, 1446  
`\numlabfont` ..... 219  
`\numpagelinesL` .....  
     .... 1808, 1864, 1875, 1879, 2002  
`\numpagelinesR` .....  
     .... 1808, 1885, 1895, 1899, 2016
- O**
- `\oldchapter` ..... 741, 750  
`\oldstanza` 767, 768, 770, 789, 790, 793  
`\one@line` ..... 942, 944, 955  
`\one@lineR` ..... 797, 979, 981, 992  
`\openout` ..... 583, 588, 591, 595
- P**
- `\p@pstartL` ..... 847  
`\p@pstartR` ..... 887  
`\page@action` ..... 285, 413, 541  
`\page@num` ..... 266, 346, 1230, 1461  
`\page@numR` ..... 239, 258, 336,  
     546, 551, 1083, 1206, 1251, 1449  
`\pagegoal` ..... 2034  
`\Pages` ..... 4, 1812  
`pages` (environment) ..... 4, 734  
`\pagetotal` ..... 1928, 2001, 2015

- pairs (environment) ..... 3, 734  
 \par@line ..... 897, 1713, 1825  
 \par@lineR ..... 797, 913, 1714, 1826  
 \pausenumbering ..... 781  
 \pausenumberingR ..... 105, 781  
 \pend ..... 5, 764, 786, 821, 1568  
 \pendL ..... 764, 889  
 \pendR ..... 786, 862, 905  
 \prevgraf .....  
     . 896, 912, 1711, 1712, 1823, 1824  
 \printlines ..... 1328  
 \printlinesR ..... 7, 1317  
 \ProcessOptions ..... 10  
 \protected@csxdef ..... 1494, 1505  
 \protected@edef ..... 846, 886  
 \protected@write ... 1343, 1355, 1661  
 \ProvidesPackage ..... 3  
 \pst@rtedLfalse ..... 32, 42  
 \pst@rtedLtrue ..... 123, 827  
 \pst@rtedRfalse ..... 34, 55, 79  
 \pst@rtedRtrue ..... 109, 131, 868  
 \pstart ..... 5, 21, 25, 762, 785, 1570  
 \pstartL ..... 762, 800  
 \pstartnumfalse ..... 1265, 1270  
 \pstartnumRfalse ..... 1277, 1282  
 \pstartnumRtrue .... 1273, 1720, 2088  
 \pstartnumtrue ..... 1719, 2059  
 \pstartR ..... 785, 800
- R**
- \Rcolwidth .. 4, 15, 746, 884, 985, 999  
 \read@linelist ..... 241, 579  
 \rem@inder ..... 1188, 1190–1192  
 \resetprevline@ 1747, 1751, 2055, 2084  
 \resumenumbering ..... 782  
 \resumenumberingR ..... 105, 782  
 \rightlinenumR ..... 212, 1201, 1209  
 \rightpstartnumL ..... 1218  
 \rightpstartnumR ..... 1218  
 Rightside (environment) ..... 5, 777  
 \Rightsidehook ..... 772, 788  
 \Rightsidehookend ..... 772, 794  
 \rlap 1201, 1209, 1225, 1233, 1246, 1254  
 \Rlineflag 7, 207, 219, 1320, 1324, 1364  
 \rule ..... 1768
- S**
- \sc@n@list ..... 1189, 1191  
 \secdef ..... 755  
 \section@num ..... 44, 46, 47, 124–126  
 \section@numR .....  
     . 30, 57, 65, 66, 110–112, 132–134  
 \select@language ... 1659, 1661–1664  
 \selectlanguage ..... 1667  
 \set@line ..... 669, 692, 713  
 \set@line@action .....  
     ..... 278, 382, 391, 398, 421, 543  
 \setl@dlp@rbox ..... 1454, 1466  
 \setl@drp@rbox ..... 1456, 1464  
 \setline ..... 634  
 \setlinenum ..... 642  
 \setnamebox ..... 835, 876, 1589  
 \setprintlines ..... 1318  
 \shiftedpstartsfalse ..... 7  
 \shiftedpstartstrue ..... 6, 8, 9  
 \shiftedversesfalse ..... 7  
 \shiftedversestrue ..... 6  
 \showlemma ..... 678, 701  
 \sidenote@margin ..... 1388, 1392  
 \sidenote@marginR ..... 1381, 1459  
 \sidenotecontent@ .....  
     1438, 1443, 1444, 1454, 1456, 1466  
 \sidenotecontent@t ..... 1464  
 \sidenotemargin ..... 1381  
 \sidenotessep ..... 1444  
 \skip@lockoff ..... 486, 494  
 \skipnumbering ..... 8, 659  
 \skipnumbering@reg ..... 663  
 \smash ..... 1768  
 \splittopskip ..... 939, 976  
 \stanza ... 767, 768, 770, 789, 790, 793  
 \stanza@count ..... 1550, 1564, 1576  
 \stanza@hang ..... 1552, 1584  
 \stanza@modulo ..... 1550, 1579  
 \stanzaindentbase .....  
     ..... 1530, 1540, 1577, 1580  
 \startlock ..... 650  
 \startstanzahook ..... 1548  
 \startsub ..... 617  
 \sub@action ..... 294, 442, 542  
 \sub@change ..... 64, 288, 289, 295  
 \sub@lock ..... 1027  
 \sub@lockR ..... 61, 303, 305, 307,  
     310, 460, 466, 467, 469, 470,  
     500, 501, 503, 1010, 1064, 1066,  
     1067, 1069, 1125, 1165, 1167, 1169  
 \sub@off ..... 625, 626  
 \sub@on ..... 619, 620  
 \subline@num ..... 221, 344, 361,  
     362, 364, 394, 434, 1028, 1033, 1377



<code>\subline@numR</code> . . . . . 222, 226, 311, 315, 334, 355, 356, 358, 387, 425, 548, 552, 1011, 1016, 1085, 1092, 1179, 1180, 1373	<b>V</b>
<code>\sublinenumincrement</code> . . . . . 5, 186	<code>\value</code> . . . . . 824, 865, 1575, 1704, 1705, 1813, 1814, 2052, 2081
<code>\sublinenumr@p</code> . 1321, 1325, 1373, 1377	<code>\vbadness</code> . . . . . 938, 975
<code>\sublinenumrepR</code> . . . . . 209, 222	<code>\vbfnoteX</code> . . . . . 1495, 1506
<code>\sublines@false</code> . . . . . 62, 292, 1054	<code>\vbox</code> . . . . . 835, 876
<code>\sublines@true</code> . . . . . 290, 1052	<code>\vl@dbfnote</code> . . . . . 1475, 1479
<code>\sublock@disp</code> . . . . . 1127, 1131, 1136	<code>\vl@dcnote</code> . . . . . 1427, 1431
<code>\symplinenum</code> . . . . . 1320	<code>\vl@dlsnote</code> . . . . . 1399, 1403
<code>\sza@penalty</code> . . . . . 1559, 1563	<code>\vl@drsnote</code> . . . . . 1413, 1417
<b>T</b>	<code>\vsplit</code> . . . . . 942, 979
<code>\textwidth</code> . . . . . 16, 18, 745, 746	<b>W</b>
<code>\theledlanguageL</code> 1652, 1667, 1730, 1859	<code>\wd</code> . . . . . 955, 992
<code>\theledlanguageR</code> 1652, 1667, 1732, 1882	<code>\writtenlinesLfalse</code> . . . . 1829, 2050
<code>\thepage</code> . . . . . 600, 1344, 1356	<code>\writtenlinesLtrue</code> . . . . . 2047
<code>\thepstart</code> . . . . . 763, 784	<code>\writtenlinesRfalse</code> . . . . 1830, 2079
<code>\thepstartL</code> . . . . .	<code>\writtenlinesRtrue</code> . . . . . 2076
. 7, 763, 803, 840, 847, 1264, 1269	<b>X</b>
<code>\thepstartR</code> . . . . .	<code>\x@lemma</code> . . . . . 680–682, 703–705
. 7, 784, 806, 880, 887, 1276, 1281	<code>\xpg@main@language</code> . . . . . 1697, 1698
<code>\thr@@</code> . . 469, 478, 501, 508, 1059, 1067	<code>\xpg@set@language</code> . . . . . 1692, 1696
<code>\topskip</code> . . . . . 1928	<code>\xright@appenditem</code> . . . . .
<b>U</b>	. . . . . 415, 416, 418, 419, 423, 430, 432, 439, 444, 446, 448, 451, 453, 455, 463, 465, 474, 497, 499, 506, 525, 526, 536, 550, 562, 566, 570, 573, 1372, 1376, 1399, 1403, 1413, 1417, 1427, 1431, 1475, 1479, 1495, 1506
<code>\unhbox</code> . . . . . 1594, 1736, 1738, 1866, 1868, 1887, 1889	<b>Z</b>
<code>\unhnamebox</code> . . . . . 1589	<code>\zz@@@</code> . . . . . 1336, 1348
<code>\unvbox</code> . . . . . 944, 981, 1596	
<code>\unvnamebox</code> . . . . . 1589	
<code>\usernamecount</code> . . . . .	
. 1551, 1558, 1600, 1632, 1838, 2042, 2060, 2062, 2071, 2089, 2091	

## Change History

v0.1	<code>\edlabel</code> commands which start a paragraph are now put in the right place. . . . . 1
General: First public release . . . . . 1	
v0.10	
General: <code>\edlabel</code> commands on the right side are now correctly indicated. . . . . 1	
v0.11	
General: Change <code>\do@lineL</code> and <code>\do@lineR</code> to allow line num-	

bering by pstart(like in eledmac 0.15). . . . .	37	<b>Leftside:</b> Added hooks into Left-side environment . . . . .	32
Lineation can be by pstart (like in eledmac 0.15). . . . .	14	<b>\flag@end:</b> Removed extraneous spaces from <b>\flag@end</b> . . . . .	28
New management of hangingsymbol insertion, preventing undesirable insertions. . . . .	52	<b>\ifledRcol:</b> Moved <b>\ifl@dpairing</b> to eledmac . . . . .	11
Prevent shift of column separator when a verse is hanged . . . . .	53	<b>\ifpst@rtedR:</b> Moved <b>\ifpst@rtedL</b> to eledmac . . . . .	12
<b>\affixline@numR:</b> Changed <b>\affixline@numR</b> to allow to disable line numbering (like in eledmac 0.15). . . . .	41	<b>\l@dlinenumR:</b> Simplified <b>\leftlinenumR</b> and <b>\rightlinenumR</b> by introducing <b>\l@dlinenumR</b> . . . . .	17
<b>\Columns:</b> Line numbering by pstart. . . . .	60	<b>\l@dnumpstartsR:</b> Moved <b>\l@dnumpstartsL</b> to eledmac . . . . .	55
<b>\get@nextboxR:</b> Change <b>\get@nextboxL</b> and <b>\get@nextboxR</b> to allow to disable line numbering (like in eledmac 0.15). . . . .	68	<b>\ledsavedprintlines:</b> Simplified <b>\printlinesR</b> by using <b>\setprintlines</b> . . . . .	47
Pstart number can be printed in side . . . . .	68	<b>\ledstrutR:</b> Added <b>\ledtrutL</b> and <b>\ledstrutR</b> . . . . .	65
v0.12		<b>\normalbfnoteX:</b> Removed extraneous spaces from <b>\normalbfnoteX</b> . . . . .	52
General: New new management of hangingsymbol insertion, preventing undesirable insertions. . . . .	52	<b>\Pages:</b> Added <b>\ledstrutL</b> to <b>\Pages</b> . . . . .	63
v0.2		Added <b>\ledstrutR</b> to <b>\Pages</b> . . . . .	64
General: Added section of babel related code . . . . .	56	<b>\Rightsidehookend:</b> Added <b>\Leftsidehook</b> , <b>\Leftsidehookend</b> , <b>\Rightsidehook</b> and <b>\Rightsidehookend</b> . . . . .	33
Fix babel problems . . . . .	1	<b>\sublinenumrepR:</b> Added <b>\linenumrepR</b> and <b>\sublinenumrepR</b> . . . . .	17
<b>\Columns:</b> Added <b>\l@checklang</b> and <b>\l@duselanguage</b> to <b>\Columns</b> . . . . .	59	v0.3.a	
<b>\Pages:</b> Added <b>\l@duselanguage</b> to <b>\Pages</b> . . . . .	63	General: Minor <b>\linenummargin</b> fix . . . . .	1
v0.3		v0.3.b	
General: Added <b>\do@lineLhook</b> and <b>\do@lineRhook</b> . . . . .	38	General: Improved parallel page balancing . . . . .	1
Reorganize for ledarab . . . . .	1	v0.3.c	
<b>\affixline@numR:</b> Changed <b>\affixline@numR</b> to match new eledmac . . . . .	41	General: Compatibilty with Polyglossia . . . . .	1
<b>\do@actions@nextR:</b> Used <b>\do@actions@fixedcode</b> in <b>\do@actionsR</b> . . . . .	40	v0.3a	
<b>\do@lineL:</b> Added <b>\do@lineLhook</b> to <b>\do@lineL</b> . . . . .	37	<b>\line@marginR:</b> Don't just set <b>\line@marginR</b> in <b>\linenummargin</b> . . . . .	15
Simplified <b>\do@lineL</b> by using macros for some common code . . . . .	37	v0.3b	
<b>\do@lineR:</b> Changed <b>\do@lineR</b> similarly to <b>\do@lineL</b> . . . . .	38	<b>\Pages:</b> Added <b>\l@dminpagelines</b> calculation for succeeding page pairs . . . . .	64

v0.4	General: No more ledparpatch. All patches are now in the main file. . . . .	1	Debug in lineation by pstart . . .	14
v0.5	General: Corrections about <code>\section</code> and other titles in numbered sections . . . . .	1	v1.0.1	General: Correction on <code>\numberonlyfirstinline</code> with lineation by pstart or by page. . . . .
v0.6	General: Be able to use <code>\chapter</code> in parallel pages. . . . .	1	v1.1	General: Shiftedverses becomes shiftedpstarts. . . . .
v0.7	General: Option ‘shiftedverses’ which make there is no blank between two parallel verses with inequal length. . . . .	1	<code>\pstartR</code> : Add <code>\labelpstarttrue</code> (from eledmac). . . . .	34
v0.8	General: Possibility to have a symbol on each hanging of verses, like in the french typography. Redefine the commande <code>\hangingsymbol</code> to define the character. . . . .	1	v1.1.1	<code>\pstartR</code> : Correct <code>\pstartR</code> bug introduced by 1.1. . . . .
v0.9	General: Possibility to number <code>\pstart</code> . . . . .	7	v1.1.2	<code>\affixside@noteR</code> : Remove spurious space between line number and line content . . . . .
	Possibility to number the pstart with the commands <code>\numberpstarttrue</code> . . . . .	1	v1.2	General: Support for <code>\led&lt;section&gt;</code> commands in parallel texts. . . .
	<code>\ifledRcol</code> : Moved <code>\iflledRcol</code> and <code>\ifnumberingR</code> to eledmac	11	v1.2.1	<code>\initnumbering@sectcountR</code> : For the right section, the counter is defined only once. . . . .
v0.9.1	General: The numbering of the pstarts restarts on each <code>\beginnumbering</code> . . . . .	1	v1.3	General: Manage RTL language. . .
v0.9.2	General: Debug : with <code>\Columns</code> , the hanging indentation now runs on the left columns and the hanging symbol is shown only when <code>\stanza</code> is used. . . . .	1	v1.3.2	General: Debug with some classes. .
v0.9.3	General: <code>\thepstartL</code> and <code>\thepstartR</code> use now <code>\bfseries</code> and not <code>\bf</code> , which is deprecated and makes conflicts with memoir class. . . . .	1	v1.3.3	<code>\l@dbfnote</code> : Spurious space with footnote in right column. . . .
v1.0	General: Compatibility with eledmac. Change name to eledpar. .	1	<code>\l@dcsnote</code> : Debug on the left notes of the right column. . .	50
			v1.3.4	<code>\l@dcsnote</code> : Allow to use commands in sidenotes, like it was introduced by eledmac 1.0. . .
			v1.3.5	<code>\normalbfnoteX</code> : Allows one to redefine <code>\thefootnoteX</code> with alph when some packages are loaded. . . . .
			v1.4	General: Added <code>\do@insidelineLhook</code> and <code>\do@insidelineRhook</code> . . .
			v1.4.1	<code>\normalbfnoteX</code> : Fix bug with normal familiar footnotes when mixing RTL and LTR text. . .

<b>astanza:</b> Enable the use of stan-	verse when a verse is exactly the
zaindentsrepetition within as-	length of a line. . . . . 1
tanza environment. . . . . 53	<b>\inserthangingsymbolR:</b> Hang
v1.4.2	verse is now not automatically
<b>\line@list@stuffR:</b> Open / close	flush right. . . . . 52
immediatly the line-list file	<b>\pendL:</b> Spurious spaces in <b>\pendL</b> . 36
when in minipage, except if the	<b>\pendR:</b> Spurious spaces in
minipage is a ledgroup. . . . . 27	<b>\pstartR</b> . . . . . 36
v1.4.3	<b>\pstartR:</b> Spurious spaces in
General: Corrects a false hanging	<b>\pstartL</b> and <b>\pstartR</b> . . . . 34