

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

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

Abstract

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

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

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

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

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

Contents

1 Introduction	3
2 The eledpar package	4
2.1 General	4
3 Parallel columns	5
4 Facing pages	6

*This file (**eledpar.dtx**) has version number v1.11.0, last revised 2015/01/23.

[†]herries dot press at earthlink dot net

[‡]maieul at maieul dot net

5 Left and right texts	7
6 Numbering text lines and paragraphs	9
7 Verse	10
8 Side notes	12
9 Parallel ledgroups	12
9.1 Parallel ledgroups and <code>setspace</code> package	14
10 Sectioning commands	14
11 Implementation overview	14
12 Preliminaries	14
12.1 Messages	15
13 Sectioning commands	16
14 Line counting	19
14.1 Choosing the system of lineation	19
14.2 Line-number counters and lists	22
14.3 Reading the line-list file	23
14.4 Commands within the line-list file	24
14.5 Writing to the line-list file	32
15 Marking text for notes	34
16 Parallel environments	35
17 Paragraph decomposition and reassembly	37
17.1 Boxes, counters, <code>\pstart</code> and <code>\pend</code>	37
17.2 Processing one line	41
17.3 Line and page number computation	44
17.4 Line number printing	47
17.5 Pstart number printing in side	49
17.6 Add insertions to the vertical list	50
17.7 Penalties	51
17.8 Printing leftover notes	52
18 Footnotes	52
18.1 Normal footnote formatting	52
19 Cross referencing	53
20 Side notes	54

<i>List of Figures</i>	3
21 Familiar footnotes	56
22 Verse	57
23 Naming macros	59
24 Counts and boxes for parallel texts	59
25 Fixing babel	61
26 Parallel columns	64
27 Parallel pages	70
28 Sections' titles' commands	80
29 Page break/no page break, depending on the specific line	81
30 Parallel ledgroup	81
31 The End	84
Appendix A Some things to do when changing version	85
Appendix A.1 Migration to eledpar 1.4.3	85
References	85
Index	85
Change History	96

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 L^AT_EX. The eledmac package was introduced in 2003 in an attempt to satisfy that request.

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

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

This manual contains a general description of how to use `eledpar` starting in section 2; the complete source code for the package, with extensive documentation (in sections 11 through 31); and an Index to the source code. As `eledpar` is an adjunct to `eledmac` I assume that you have read the `eledmac` manual. Also `eledpar` requires `eledmac` to be used, preferably at least version 0.10 (2011/08/22). You do not need to read the source code for this package in order to use it but doing so may help to answer any questions you might have. On a first reading, I suggest that you should skip anything after the general documentation in sections 2 until 11, unless you are particularly interested in the innards of `eledpar`.

2 The `eledpar` package

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

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

2.1 General

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

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

both the number of potential boxes and memory are limited. If TeX's memory is overfilled the recourse is to reduce the amount of text stored before printing.

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

```
\maxchunks{5120}
```

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

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

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

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

3 Parallel columns

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

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

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

Keep in mind that the `\Columns` **must be** outside of the `pairs` environment.

`\AtBeginPairs` You can use the macro `\AtBeginPairs` to insert a code at the beginning of each `pairs` environments. That could be useful to add the `\sloppy` macro to prevent overfull hboxes in two columns.

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

	There is no required pagebreak before or after the columns.
<code>\Lcolwidth</code>	The lengths <code>\Lcolwidth</code> and <code>\Rcolwidth</code> are the widths of the left and right columns, respectively. By default, these are: <code>\setlength{\Lcolwidth}{0.45\textwidth}</code> <code>\setlength{\Rcolwidth}{0.45\textwidth}</code>
<code>\Rcolwidth</code>	
	They may be adjusted if one text tends to be ‘bulkier’ than the other.
<code>\columnrulewidth</code>	The macro <code>\columnseparator</code> is called between each left/right pair of lines. By default it inserts a vertical rule of width <code>\columnrulewidth</code> . As this is initially defined to be 0pt the rule is invisible. For a visible rule between the columns you could try: <code>\setlength{\columnrulewidth}{0.4pt}</code>
<code>\columnseparator</code>	
	You can also modify <code>\columnseparator</code> if you want more control.
<code>\columnspan</code>	By default, columns are positioned to the right of the page. However, you use <code>\columnspan{L}</code> to align them to the left, or <code>\columnspan{C}</code> to center them.
	When you use <code>\stanza</code> , the visible rule may shift when a verse has a hanging indent. To prevent shifting, use <code>\setstanzaindents</code> outside the <code>Leftside</code> or <code>Rightside</code> environment.
<code>\beforecolumnseparator</code>	By default, the spaces around column separator are the same as the space: <ul style="list-style-type: none"> • On the left of columns, if columns are aligned right. • On the right of columns, if columns are aligned left. • On both the Left and Right columns, if columns are centered.
<code>\aftercolumnseparator</code>	
	You can redefine <code>\beforecolumnseparator</code> and <code>\aftercolumnseparator</code> length to define spaces before or after the column separator, instead of letting <code>eledpar</code> calculate them automatically. <code>\setlength{\beforecolumnseparator}{length}</code> <code>\setlength{\aftercolumnseparator}{length}</code>
<code>\widthliketwocolumns</code>	If you want to come back to the previous behavior, just set them with a negative value. If you want to mix texts in columns and text without columns, you can horizontally align text in one column to text in two columns with <code>\widthliketwocolumnstrue</code> . To reset this feature, just use <code>\widthliketwocolumnsfalse</code> .
<code>\Xnoteswidthliketwocolumns</code>	In most case, you should use <code>\widthliketwocolumns</code> in combination with <code>\Xnoteswidthliketwocolumns</code> and <code>\notesXwidthliketwocolumns</code> to align the critical/familiar footnotes with the two columns. <code>eledmac</code> handbook for more details.
<code>\notesXwidthliketwocolumns</code>	

4 Facing pages

`pages` Numbered text that is to be set on facing pages must be within a `pages` environ-

ment. Within the environment the text for the lefthand and righthand pages is placed within the `Leftside` and `Rightside` environments, respectively.

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

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

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

`\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.

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`¹ 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

¹when used with `ledpatch` v0.2 or greater.

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

gives you independent control over the left and right numbering schemes. The `\firstsublinenum` and `\sublinenumincrement` macros correspondingly set the numbering scheme for sublines. The starred versions change both left and right numbering schemes.

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

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

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

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

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

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

6 Numbering text lines and paragraphs

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

```
\beginnumbering
<text>
```

```
\endnumbering
```

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

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

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

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

`\Rlineflag` The value of `\Rlineflag` is appended to the line numbers of the right texts.

Its default definition is:

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

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

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

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

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

7 Verse

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

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

If you get an error message along the lines of ‘Missing number, treated as zero `\sza000`’ it is because you have forgotten to use `\setstanzaindent` to set the stanza indents.

```
\skipnumbering The command \skipnumbering when inserted in a line of parallel text causes
the numbering of that particular line to be skipped. This can 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 hanging line. If you use it, you must run \LaTeX two time. Example for the French typography

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

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

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

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

8 Side notes

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

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

9 Parallel ledgroups

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

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

See the following example:

```

\begin{pages}
\begin{Leftside}
\beginnumbering
\pstart
\begin{ledgroup}
  ledgroup content
\end{ledgroup}

```

```

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

```

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

```

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

```

9.1 Parallel ledgroups and `setspace` package

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

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

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

10 Sectioning commands

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

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

11 Implementation overview

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

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

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

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

12 Preliminaries

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

```

1 (*code)
2 \NeedsTeXFormat{LaTeX2e}
3 \ProvidesPackage{eledpar}[2015/01/23 v1.11.0 eledmac extension for parallel texts]%
4

```

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

`\ifshiftedpstarts`

```

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

```

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

```

11 \DeclareOption{widthliketwocolumns}{\widthliketwocolumnstrue}%

```

```

12 \ProcessOptions%

```

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

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

```

13 \l@dpairingfalse
14 \l@dpagingfalse
15 \ledRcolfalse

```

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

```

\ Rcolwidth
16 \newdimen\Lcolwidth
17 \Lcolwidth=0.45\textwidth
18 \newdimen\Rcolwidth
19 \Rcolwidth=0.45\textwidth
20

```

12.1 Messages

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

`\eledpar@error`

```
21 \newcommand{\eledpar@error}[2]{\PackageError{eledpar}{#1}{#2}}
22 % \end{macrocode}
23 % \end{macro}
24 % \begin{macro}{\led@err@TooManyPstarts}
25 % \begin{macrocode}
26 \newcommand*{\led@err@TooManyPstarts}{%
27 \eledpar@error{Too many \string\pstart\space without printing.
28 \qquad Some text will be lost}{\@ehc}}
```

`\led@err@BadLeftRightPstarts`

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

`\led@err@LeftOnRightPage`

`\led@err@RightOnLeftPage`

```
32 \newcommand*{\led@err@LeftOnRightPage}{%
33 \eledpar@error{The left page has ended on a right page}{\@ehc}}
34 \newcommand*{\led@err@RightOnLeftPage}{%
35 \eledpar@error{The right page has ended on a left page}{\@ehc}}
```

13 Sectioning commands

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

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

```
36 \newcount\section@numR
37 \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`.

```
38 \pst@rtedLfalse
39 \newif\ifpst@rtedR
40 \pst@rtedRfalse
41
```

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

```
42 \newcommand*{\beginnumberingR}{%
43 \ifnumberingR
44 \led@err@NumberingStarted
45 \endnumberingR
46 \fi
```



```

47 \global\l@dumpstartsR \z@
48 \global\pst@rtedRfalse
49 \global\numberingRtrue
50 \global\advance\section@numR \@ne
51 \global\absline@numR \z@
52 \gdef\normal@page@breakR{ }
53 \gdef\l@prev@pbR{ }
54 \gdef\l@prev@nopbR{ }
55 \global\line@numR \z@
56 \global\@lockR \z@
57 \global\sub@lockR \z@
58 \global\sublines@false
59 \global\let\next@page@numR\relax
60 \global\let\sub@change\relax
61 \message{Section \the\section@numR R }%
62 \line@list@stuffR{\jobname.\extensionchars\the\section@numR R}%
63 \l@dend@stuff
64 \setcounter{pstartR}{1}
65 \begingroup
66 \initnumbering@sectcountR
67 \gdef\eled@sectionsR@{ }%
68 \if@noeled@sec\else%
69 \makeatletter\inputIfFileExists{\jobname.eledsec\the\section@numR R}{ }{\makeatother}%
70 \immediate\openout\eled@sectioningR@out=\jobname.eledsec\the\section@numR R\relax%
71 \fi%
72 }

```

\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.

```

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

```

```

90 \else
91   \led@err@NumberingNotStarted
92 \fi
93 \endgroup
94 \if@noeled@sec\else%
95   \immediate\closeout\eled@sectioningR@out%
96 \fi%
97 }
98

```

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

```

99 \newcounter{chapterR}
100 \newcounter{sectionR}
101 \newcounter{subsectionR}
102 \newcounter{subsubsectionR}
103 \newcommand{\initnumbering@sectcountR}{
104   \let\c@chapter\c@chapterR
105   \let\c@section\c@sectionR
106   \let\c@subsection\c@subsectionR
107   \let\c@subsubsection\c@subsubsectionR
108 }

```

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

```

109 \newcommand*{\pausenumberingR}{%
110   \endnumberingR\global\numberingRtrue}
111 \newcommand*{\resumenumberingR}{%
112   \ifnumberingR
113     \global\pst@rtedRtrue
114     \global\advance\section@numR \@ne
115     \led@mess@SectionContinued{\the\section@numR R}%
116     \line@list@stuffR{\jobname.\extensionchars\the\section@numR R}%
117     \l@dend@stuff
118     \begingroup%
119     \initnumbering@sectcountR%
120   \else
121     \led@err@numberingShouldHaveStarted
122     \endnumberingR
123     \beginnumberingR
124   \fi}
125

```

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

```

126 \newcommand*{\memorydumpL}{%
127   \endnumbering
128   \numberingtrue

```

```

129 \global\pst@rtedLtrue
130 \global\advance\section@num \@ne
131 \led@mess@SectionContinued{\the\section@num}%
132 \line@list@stuff{\jobname.\extensionchars\the\section@num}%
133 \l@dend@stuff}
134 \newcommand*{\memorydumpR}{%
135 \endnumberingR
136 \numberingRtrue
137 \global\pst@rtedRtrue
138 \global\advance\section@numR \@ne
139 \led@mess@SectionContinued{\the\section@numR R}%
140 \line@list@stuffR{\jobname.\extensionchars\the\section@numR R}%
141 \l@dend@stuff}
142

```

14 Line counting

14.1 Choosing the system of lineation

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

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

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

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

`\ifbypage@R`

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

`\bypage@Rfalse`

```

143 \newif\ifbypage@R
144 \newif\ifbypstart@R
145 \bypage@Rfalse
146 \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`.

```

147 \newcommand*{\lineationR}[1]{%
148 \ifnumbering
149 \led@err@LineationInNumbered
150 \else
151 \def\@tempa{#1}\def\@tempb{page}%
152 \ifx\@tempa\@tempb
153 \global\bypage@Rtrue
154 \global\bypstart@Rfalse
155 \else
156 \def\@tempb{pstart}%
157 \ifx\@tempa\@tempb

```

```

158         \global\bypage@Rfalse
159         \global\bystart@Rtrue
160     \else
161         \def@tempb{section}
162         \ifx\@tempa\@tempb
163             \global\bypage@Rfalse
164             \global\bystart@Rfalse
165         \else
166             \led@warn@BadLineation
167         \fi
168     \fi
169 \fi
170 \fi}}

```

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

```

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

```

`\linenummargin` You call `\linenummargin{<word>}` to specify which margin you want your right text's line numbers in; it takes one argument, a string. You can put the line numbers in the same margin on every page using `left` or `right`; or you can use `inner` or `outer` to get them in the inner or outer margins. You can change this within a numbered section, but the change may not take effect just when you'd like; if it's done between paragraphs nothing surprising should happen.

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

```

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

```

By default put right text numbers at the right.

```

185 \line@marginR=\@ne
186

```

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

```

187 \newcounter{firstlinenumR}

```

```

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

```

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

```

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

```

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

```

\sublinenumincrement 196 \providecommand*\firstlinenum{}
\firstlinenum*       197 \providecommand*\linenumincrement{}
\linenumincrement*   198 \providecommand*\firstsublinenum{}
\firstsublinenum*    199 \providecommand*\sublinenumincrement{}
\sublinenumincrement* 200 \renewcommand*\firstlinenum[1]{%
201   \ifledRcol \setcounter{firstlinenumR}{#1}%
202   \else      \setcounter{firstlinenumR}{#1}%
203   \fi}
204 \renewcommand*\linenumincrement[1]{%
205   \ifledRcol \setcounter{linenumincrementR}{#1}%
206   \else      \setcounter{linenumincrementR}{#1}%
207   \fi}
208 \renewcommand*\firstsublinenum[1]{%
209   \ifledRcol \setcounter{firstsublinenumR}{#1}%
210   \else      \setcounter{firstsublinenumR}{#1}%
211   \fi}
212 \renewcommand*\sublinenumincrement[1]{%
213   \ifledRcol \setcounter{sublinenumincrementR}{#1}%
214   \else      \setcounter{sublinenumincrementR}{#1}%
215   \fi}
216 \WithSuffix\newcommand\firstlinenum*[1]{\setcounter{firstlinenumR}{#1}\setcounter{firstlinenum}{#1}}
217 \WithSuffix\newcommand\linenumincrement*[1]{\setcounter{linenumincrementR}{#1}\setcounter{linenumincr
218 \WithSuffix\newcommand\firstsublinenum*[1]{\setcounter{subfirstlinenumR}{#1}\setcounter{subfirstlinen
219 \WithSuffix\newcommand\sublinenumincrement*[1]{\setcounter{sublinenumincrementR}{#1}\setcounter{subli

```

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

```

220 \newcommand*\Rlineflag{R}
221

```

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

```

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

```

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

```

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

14.2 Line-number counters and lists

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

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

```

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

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

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

```

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

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

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

```

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

```

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

```

254 \newcount\page@numR
255

```

14.3 Reading the line-list file

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

```

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

```

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

```

257   \ifledRcol
258     \list@clear{\line@listR}%
259     \list@clear{\insertlines@listR}%
260     \list@clear{\actionlines@listR}%
261     \list@clear{\actions@listR}%
262     \list@clear{\linesinpar@listR}%
263     \list@clear{\linesonpage@listR}
264     \list@clear{\sw@listR}%
265     \list@clear{\sw@list@inedtextR}%
266   \else
267     \list@clearing@reg
268     \list@clear{\linesinpar@listL}%
269     \list@clear{\linesonpage@listL}%
270   \fi

```

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

```

271   \list@clear{\maxlinesinpar@list}

```

Now get the file and interpret it.

```

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

```

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

```

274   \ifledRcol
275     \global\page@numR=\m@ne
276     \ifx\actionlines@listR\empty
277       \gdef\next@actionlineR{1000000}%
278     \else
279       \gl@p\actionlines@listR\to\next@actionlineR

```

```

280     \glp\actions@listR\to\next@actionR
281   \fi
282 \else
283   \global\page@num=\m@ne
284   \ifx\actionlines@list\empty
285     \gdef\next@actionline{1000000}%
286   \else
287     \glp\actionlines@list\to\next@actionline
288     \glp\actions@list\to\next@action
289   \fi
290 \fi}
291

```

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

14.4 Commands within the line-list file

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

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

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

```

292 \newcommand{\@nl@regR}{%
293   \ifx\l@dchset@num\relax \else
294     \advance\absline@numR \@ne
295     \set@line@action
296     \let\l@dchset@num\relax
297     \advance\absline@numR \m@ne
298     \advance\line@numR \m@ne%    % do we need this?
299   \fi
300   \advance\absline@numR \@ne
301   \ifx\next@page@numR\relax \else
302     \page@action
303     \let\next@page@numR\relax
304   \fi
305   \ifx\sub@change\relax \else
306     \ifnum\sub@change>z@
307       \sublines@true
308     \else
309       \sublines@false
310     \fi
311     \sub@action
312     \let\sub@change\relax
313   \fi

```



```

314 \ifcase\@lockR
315 \or
316   \@lockR \tw@
317 \or\or
318   \@lockR \z@
319 \fi
320 \ifcase\sub@lockR
321 \or
322   \sub@lockR \tw@
323 \or\or
324   \sub@lockR \z@
325 \fi
326 \ifsublines@
327   \ifnum\sub@lockR<\tw@
328     \advance\subline@numR \@ne
329   \fi
330 \else
331   \ifnum\@lockR<\tw@
332     \advance\line@numR \@ne \subline@numR \z@
333   \fi
334 \fi}
335
336 \renewcommand*{\@nl}[2]{%
337   \fix@page{#1}%
338   \ifledRcol
339     \@nl@regR
340   \else
341     \@nl@reg
342   \fi}
343

```

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

```

\fix@page 344 \newcount\last@page@numR
345   \last@page@numR=-10000
346 \renewcommand*{\fix@page}[1]{%
347   \ifledRcol
348     \ifnum #1=\last@page@numR
349     \else
350       \ifbypage@R
351         \line@numR \z@ \subline@numR \z@
352       \fi
353       \page@numR=#1\relax
354       \last@page@numR=#1\relax
355       \def\next@page@numR{#1}%
356     \fi
357   \else
358     \ifnum #1=\last@page@num
359     \else
360       \ifbypage@
361         \line@num \z@ \subline@num \z@

```

```

362     \fi
363     \page@num=#1\relax
364     \last@page@num=#1\relax
365     \def\next@page@num{#1}%
366     \listxadd{\normal@page@break}{\the\absline@num}
367     \fi
368 \fi}
369

```

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

```

370 \renewcommand*{\@adv}[1]{%
371   \ifsublines@
372     \ifledRcol
373       \advance\subline@numR by #1\relax
374       \ifnum\subline@numR<\z@
375         \led@warn@BadAdvancelineSubline
376         \subline@numR \z@
377       \fi
378     \else
379       \advance\subline@num by #1\relax
380       \ifnum\subline@num<\z@
381         \led@warn@BadAdvancelineSubline
382         \subline@num \z@
383       \fi
384     \fi
385   \else
386     \ifledRcol
387       \advance\line@numR by #1\relax
388       \ifnum\line@numR<\z@
389         \led@warn@BadAdvancelineLine
390         \line@numR \z@
391       \fi
392     \else
393       \advance\line@num by #1\relax
394       \ifnum\line@num<\z@
395         \led@warn@BadAdvancelineLine
396         \line@num \z@
397       \fi
398     \fi
399   \fi
400   \set@line@action}
401

```

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

```

402 \renewcommand*{\@set}[1]{%
403   \ifledRcol
404     \ifsublines@

```

```

405     \subline@numR=#1\relax
406   \else
407     \line@numR=#1\relax
408   \fi
409   \set@line@action
410 \else
411   \ifsublines@
412     \subline@num=#1\relax
413   \else
414     \line@num=#1\relax
415   \fi
416   \set@line@action
417 \fi}
418

```

`\l@d@set` The `\l@d@set{<num>}` macro sets the line number for the next `\pstart...` to the value specified as its argument. This is used to implement `\setlinenum`.
`\l@dchset@num` `\l@dchset@num` is a flag to the `\@l` macro. If it is not `\relax` then a linenum change is to be done.

```

419 \renewcommand*{\l@d@set}[1]{%
420   \ifledRcol
421     \line@numR=#1\relax
422     \advance\line@numR \@ne
423     \def\l@dchset@num{#1}
424   \else
425     \line@num=#1\relax
426     \advance\line@num \@ne
427     \def\l@dchset@num{#1}
428   \fi}
429 \let\l@dchset@num\relax
430

```

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

```

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

```

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

```

439 \renewcommand*{\set@line@action}{%
440   \ifledRcol
441     \xright@appenditem{\the\absline@numR}\to\actionlines@listR
442   \ifsublines@
443     \l@tempcnta=-\subline@numR

```

```

444 \else
445 \l@ldtempcnta=-\line@numR
446 \fi
447 \advance\l@ldtempcnta by -5000\relax
448 \xright@appenditem{\the\l@ldtempcnta}\to\actions@listR
449 \else
450 \xright@appenditem{\the\absline@num}\to\actionlines@list
451 \ifsublines@
452 \l@ldtempcnta=-\subline@num
453 \else
454 \l@ldtempcnta=-\line@num
455 \fi
456 \advance\l@ldtempcnta by -5000\relax
457 \xright@appenditem{\the\l@ldtempcnta}\to\actions@list
458 \fi}
459

```

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

```

460 \renewcommand*{\sub@action}{%
461 \ifledRcol
462 \xright@appenditem{\the\absline@numR}\to\actionlines@listR
463 \ifsublines@
464 \xright@appenditem{-1001}\to\actions@listR
465 \else
466 \xright@appenditem{-1002}\to\actions@listR
467 \fi
468 \else
469 \xright@appenditem{\the\absline@num}\to\actionlines@list
470 \ifsublines@
471 \xright@appenditem{-1001}\to\actions@list
472 \else
473 \xright@appenditem{-1002}\to\actions@list
474 \fi
475 \fi}
476

```

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

```

477 \newcount\@lockR
478 \newcount\sub@lockR
479
480 \newcommand*{\do@lockonR}{%
481 \xright@appenditem{\the\absline@numR}\to\actionlines@listR
482 \ifsublines@
483 \xright@appenditem{-1005}\to\actions@listR
484 \ifnum\sub@lockR=\z@
485 \sub@lockR \@ne

```

```

486     \else
487         \ifnum\sub@lockR=\thr@@
488             \sub@lockR \@ne
489         \fi
490     \fi
491 \else
492     \xright@appenditem{-1003}\to\actions@listR
493     \ifnum\@lockR=\z@
494         \@lockR \@ne
495     \else
496         \ifnum\@lockR=\thr@@
497             \@lockR \@ne
498         \fi
499     \fi
500 \fi}
501
502 \renewcommand*{\do@lockon}{%
503     \ifx\next\lock@off
504         \global\let\lock@off=\skip@lockoff
505     \else
506         \ifledRcol
507             \do@lockonR
508         \else
509             \do@lockonL
510         \fi
511     \fi}

```

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

```

\do@lockoff 512
\do@lockoffR 513
\skip@lockoff 514 \newcommand{\do@lockoffR}{%
515     \xright@appenditem{\the\absline@numR}\to\actionlines@listR
516     \ifsublines@
517         \xright@appenditem{-1006}\to\actions@listR
518         \ifnum\sub@lockR=\tw@
519             \sub@lockR \thr@@
520         \else
521             \sub@lockR \z@
522         \fi
523     \else
524         \xright@appenditem{-1004}\to\actions@listR
525         \ifnum\@lockR=\tw@
526             \@lockR \thr@@
527         \else
528             \@lockR \z@
529         \fi
530     \fi}
531
532 \renewcommand*{\do@lockoff}{%
533     \ifledRcol

```

```

534 \do@lockoffR
535 \else
536 \do@lockoffL
537 \fi}
538 \global\let\lock@off=\do@lockoff
539

```

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

```

540 \providecommand*\n@num-{}
541 \renewcommand*\n@num-{}%
542 \ifledRcol
543 \xright@appenditem{\the\absline@numR}\to\actionlines@listR
544 \xright@appenditem{-1007}\to\actions@listR
545 \else
546 \n@num@reg
547 \fi}
548

```

`\@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`.

```

549 \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.

```

550 \renewcommand*\@ref}[2]{%
551 \ifledRcol
552 \global\insert@countR=#1\relax
553 \loop\ifnum\insert@countR>\z@
554 \xright@appenditem{\the\absline@numR}\to\insertlines@listR
555 \global\advance\insert@countR \m@ne
556 \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.

```

557 \begingroup
558 \let\@ref=\dummy@ref

```

```

559 \let\@lopR\@gobble
560 \let\page@action=\relax
561 \let\sub@action=\relax
562 \let\set@line@action=\relax
563 \let\@lab=\relax
564 \let\@sw\@gobbletwo%
565 #2
566 \global\endpage@num=\page@numR
567 \global\endline@num=\line@numR
568 \global\endsubline@num=\subline@numR
569 \endgroup

```

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

```

570 \xright@appenditem%
571 {\the\page@numR|\the\line@numR|}%
572 \ifsublines@ \the\subline@numR \else 0\fi}%
573 \the\endpage@num|\the\endline@num|}%
574 \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.

```

575 #2
576 \else

```

And when not in right text

```

577 \@ref@reg{#1}{#2}%
578 \fi}

```

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

```

579 \providecommand*\@pend}[1]{}
580 \renewcommand*\@pend}[1]{%
581 \ifbypstart@\global\line@num=0\fi%
582 \xright@appenditem{#1}\to\linesinpar@listL}
583 \providecommand*\@pendR}[1]{}
584 \renewcommand*\@pendR}[1]{%
585 \ifbypstart@R\global\line@numR=0\fi
586 \xright@appenditem{#1}\to\linesinpar@listR}
587

```

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

```

588 \providecommand*\@lopL}[1]{}
589 \renewcommand*\@lopL}[1]{%
590 \xright@appenditem{#1}\to\linesonpage@listL}
591 \providecommand*\@lopR}[1]{}

```

```

592 \renewcommand*{\@lopR}[1]{%
593   \xright@appenditem{#1}\to\linesonpage@listR}
594

```

14.5 Writing to the line-list file

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

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

```

595 \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 596 \newif\iffirst@linenum@out@R
597   \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.

```

598 \newcommand*{\line@list@stuffR}[1]{%
599   \read@linelist{#1}%
600   \iffirst@linenum@out@R
601     \immediate\closeout\linenum@outR
602     \global\first@linenum@out@Rfalse
603     \immediate\openout\linenum@outR=#1%
604   \else
605     \if@minipage%
606       \leavevmode%
607     \fi%
608     \closeout\linenum@outR%
609     \openout\linenum@outR=#1%
610   \fi}
611

```

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

```

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

```

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

```

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

```

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

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

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

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

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

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

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

`\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.

```

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

```

`\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.

```

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

```

15 Marking text for notes

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

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

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

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

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

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

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

```

665 \renewcommand*{\set@line}{%
666   \ifledRcol
667     \ifx\line@listR\empty
668       \global\noteschanged@true
669       \xdef\l@d@nums{000|000|000|000|000|000|\edfont@info}%
670     \else
671       \glp\line@listR\to\@tempb

```

```

672     \xdef\l@d@nums{\@tempb|\edfont@info}%
673     \global\let\@tempb=\undefined
674   \fi
675 \else
676   \ifx\line@list\empty
677     \global\noteschanged@true
678     \xdef\l@d@nums{000|000|000|000|000|000|\edfont@info}%
679   \else
680     \glp\line@list\to\@tempb
681     \xdef\l@d@nums{\@tempb|\edfont@info}%
682     \global\let\@tempb=\undefined
683   \fi
684 \fi}
685

```

16 Parallel environments

The initial set up for parallel processing is deceptively simple.

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

```

chapterinpages 686 \newenvironment{pairs}{%
687   \l@dpairingtrue
688   \l@dpagingfalse
689   \initnumbering@sectcmd
690   \at@begin@pairs%
691 }{%
692   \l@dpairingfalse
693 }
694

```

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

```

695 \newcommand{\AtBeginPairs}[1]{\xdef\at@begin@pairs{#1}}%
696 \def\at@begin@pairs{}%
697

```

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.

```

698 \newenvironment{pages}{%
699   \let\oldchapter\chapter
700   \let\chapter\chapterinpages
701   \l@dpairingtrue
702   \l@dpagingtrue
703   \initnumbering@sectcmd
704   \setlength{\Lcolwidth}{\textwidth}%

```

```

705 \setlength{\Rcolwidth}{\textwidth}%
706 }{%
707 \l@dpairingfalse
708 \l@dpagingfalse
709 \let\chapter\oldchapter
710 }
711 \newcommand{\chapterinpages}{\thispagestyle{plain}%
712 \global\@topnum\z@
713 \afterindentfalse
714 \secdef\@chapter\@schapter}
715

```

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

```

716 \newif\ifinstanzaL
717 \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.

```

718 \newenvironment{Leftside}{%
719 \ledRcolfalse
720 \setcounter{pstartL}{1}
721 \let\pstart\pstartL
722 \let\thepstart\thepstartL
723 \let\pend\pendL
724 \let\memorydump\memorydumpL
725 \Leftsidehook
726 \let\old@startstanza\@startstanza
727 \def\@startstanza[##1]{\global\instanzaLtrue\old@startstanza[##1]}
728 }{
729 \Leftsidehookend}

```

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

```

\Rightsidehook 730 \newcommand*\Leftsidehook{}
\Rightsidehookend 731 \newcommand*\Leftsidehookend{}
732 \newcommand*\Rightsidehook{}
733 \newcommand*\Rightsidehookend{}
734

```

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.

```

735 \newenvironment{Rightside}{%
736 \ledRcoltrue
737 \let\beginnumbering\beginnumberingR
738 \let\endnumbering\endnumberingR

```

```

739 \let\pausenumbering\pausenumberingR
740 \let\resumenumbering\resumenumberingR
741 \let\memorydump\memorydumpR
742 \let\thepstart\thepstartR
743 \let\pstart\pstartR
744 \let\pend\pendR
745 \let\ledpb\ledpbR
746 \let\lednopb\lednopbR
747 \let\lineation\lineationR
748 \Rightsidehook
749 \let\old@startstanza\@startstanza
750 \def\@startstanza[##1]{\global\instanzaRtrue\old@startstanza[##1]}
751 }{\%
752 \ledRcolfalse
753 \Rightsidehookend
754 }
755

```

17 Paragraph decomposition and reassembly

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

17.1 Boxes, counters, \pstart and \pend

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

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

```

756 \newcount\num@linesR
757 \newbox\one@lineR
758 \newcount\par@lineR

```

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

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

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

```

759
760 \newcounter{pstartL}
761 \newcounter{pstartLold}
762 \renewcommand{\thepstartL}{\bfseries\@arabic\c@pstartL}. }
763 \newcounter{pstartR}
764 \newcounter{pstartRold}
765 \renewcommand{\thepstartR}{\bfseries\@arabic\c@pstartR}. }
766
767 \newcommandx*{\pstartL}[1][1]{%
768   \if@nobreak%
769     \let\@oldnobreak\@nobreaktrue%
770   \else%
771     \let\@oldnobreak\@nobreakfalse%
772   \fi%
773   \@nobreaktrue%
774   \ifnumbering \else%
775     \led@err@PstartNotNumbered%
776     \beginnumbering%
777   \fi%
778   \ifnumberedpar%
779     \led@err@PstartInPstart%
780   \pend%
781 \fi%
```

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

```

782 \ifpstart@rtedL\else%
783   \setcounter{pstartLold}{\value{pstartL}}%
784   \list@clear{\inserts@list}%
785   \global\let\next@insert=\empty%
786   \global\pstart@rtedLtrue%
787 \fi%
788 \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.

```

789 \global\advance\l@dnumpstartsL \@ne%
790 \ifnum\l@dnumpstartsL>\l@dc@maxchunks%
791   \led@err@TooManyPstarts%
792   \global\l@dnumpstartsL=\l@dc@maxchunks%
793 \fi%
794 \global\setnamebox{\l@dc@colrawbox\the\l@dnumpstartsL}=\vbox\bgroup%
```

```

795 \ifautopar\else%
796   \ifnumberpstart%
797     \ifsidepstartnum%
798       \else%
799         \thepstartL%
800       \fi%
801     \fi%
802   \fi%
803 \hsize=\Lcolwidth%
804 \numberedpar@true%
805 \iflabelpstart\protected@edef\@currentlabel%
806   {\p@pstartL\thepstartL}\fi%
  Dump the optional arguments
807 \ifstrempy{#1}%
808   {\csgdef{before@pstartL@the\l@dnumpstartsL}{\at@every@pstart}}%
809   {\csgdef{before@pstartL@the\l@dnumpstartsL}{\noindent#1}}%
810 }
811 \newcommandx*{\pstartR}[1][1]{%
812   \if@nobreak%
813     \let\@oldnobreak\@nobreaktrue%
814   \else%
815     \let\@oldnobreak\@nobreakfalse%
816   \fi%
817   \@nobreaktrue%
818   \ifnumberingR \else%
819     \led@err@PstartNotNumbered%
820     \beginnumberingR%
821   \fi%
822   \ifnumberedpar@%
823     \led@err@PstartInPstart%
824   \pendR%
825   \fi%
826   \ifpst@rtedR\else%
827     \setcounter{pstartRold}{\value{pstartR}}%
828     \list@clear{\inserts@listR}%
829     \global\let\next@insertR=\empty%
830     \global\pst@rtedRtrue%
831   \fi%
832   \begingroup\normal@pars%
833   \global\advance\l@dnumpstartsR \@ne%
834   \ifnum\l@dnumpstartsR>\l@dc@maxchunks%
835     \led@err@TooManyPstarts%
836     \global\l@dnumpstartsR=\l@dc@maxchunks%
837   \fi%
838   \global\setnamebox{l@dRcolrawbox\the\l@dnumpstartsR}=\vbox\bgroup%
839   \ifautopar\else%
840     \ifnumberpstart%
841       \ifsidepstartnum\else%
842         \thepstartR%

```

```

843      \fi%
844      \fi%
845      \fi%
846      \hsize=\Rcolwidth%
847      \numberedpar@true%
848      \iflabelpstart\protected@edef\@currentlabel%
849        {\p@pstartR\thepstartR}\fi%
850      \ifstrempy{#1}%
851        {\csgdef{before@pstartR@the\l@dnumpstartsR}{\at@every@pstart}}%
852        {\csgdef{before@pstartR@the\l@dnumpstartsR}{\noindent#1}}%
853    }

```

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

```

854 \newcommandx*{\pendL}[1][1]{%
855   \ifnumbering \else%
856     \led@err@PendNotNumbered%
857   \fi%
858   \ifnumberedpar@ \else%
859     \led@err@PendNoPstart%
860   \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.

```

861   \l@dzeropenalties%
862   \endgraf\global\num@lines=\prevgraf\egroup%
863   \global\par@line=0%

```

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

```

864   \endgroup%
865   \ignorespaces%
866   \@oldnobreak%
867   \ifnumberpstart%
868     \addtocounter{pstartL}{1}%
869   \fi
870   \parledgroup@beforenotes@save{L}%

```

Dump content of the optional argument.

```

871   \ifstrempy{#1}%
872     {\csgdef{after@pendL@the\l@dnumpstartsL}{\at@every@pend}}%
873     {\csgdef{after@pendL@the\l@dnumpstartsL}{\noindent#1}}%
874   }

```

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

```

875 \newcommandx*{\pendR}[1][1]{%
876   \ifnumberingR \else%
877     \led@err@PendNotNumbered%
878   \fi%

```



```

879 \ifnumberedpar@ \else%
880   \led@err@PendNoPstart%
881 \fi%
882 \l@dzeropenalties%
883 \endgraf\global\num@linesR=\prevgraf\egroup%
884 \global\par@lineR=0%
885 \endgroup%
886 \ignorespaces%
887 \@oldnobreak%
888 \ifnumberpstart%
889   \addtocounter{pstartR}{1}%
890 \fi%
891 \parledgroup@beforenotes@save{R}%
892 \ifstrempy{#1}%
893   {\csgdef{after@pendR@the\l@dnumpstartsR}{\at@every@pend}}%
894   {\csgdef{after@pendR@the\l@dnumpstartsR}{\noindent#1}}%
895 }
896

```

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

```

897 \newif\ifprint@last@after@pendL%
898 \newif\ifprint@last@after@pendR%

```

17.2 Processing one line

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

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

```

899 \newbox\l@dleftbox
900 \newbox\l@drightbox
901

```

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

```

\countRline 902 \newcount\countLline
903   \countLline \z@
904 \newcount\countRline
905   \countRline \z@
906

```

`\@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 907 \newcount\@donereallinesL
908 \newcount\@donetotallinesL
909 \newcount\@donereallinesR
910 \newcount\@donetotallinesR

```

911

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

```

912 \newcommand*{\do@lineL}{%
913   \advance\countLline \@ne
914   \ifvbox\namebox{\l@dLcolrawbox\the\l@dpscL}%
915   {\vbadness=10000
916     \splittopskip=\z@
917     \do@lineLhook
918     \l@demptyd@ta
919     \global\setbox\one@line=\vsplit\namebox{\l@dLcolrawbox\the\l@dpscL}
920       to\baselineskip}%
921   \IfStrEq{\splitfirstmarks\parledgroup@}{begin}{\parledgroup@notes@startL}{-}
922   \unvbox\one@line \global\setbox\one@line=\lastbox
923   \getline@numL
924   \ifnum\@lock>\@ne%
925     \inserthangingsymboltrue%
926   \else%
927     \inserthangingsymbolfalse%
928   \fi
929   \setbox\l@dleftbox
930   \hb@xt@ \Lcolwidth{%
931     \affixline@num
932     \xifinlist{\the\l@dpscL}{\eled@sections@@}%
933     {\add@inserts\affixside@note}%
934     {\print@lineL}}%
935   \add@penaltiesL
936   \global\advance\@donereallinesL\@ne
937   \global\advance\@donetotallinesL\@ne
938 \else
939   \setbox\l@dleftbox \hb@xt@ \Lcolwidth{\hspace*{\Lcolwidth}}%
940   \global\advance\@donetotallinesL\@ne
941 \fi}
942
943
```

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

```

944 \def\print@lineL{%
945   \affixpstart@numL%
946   \l@dld@ta %space kept for backward compatibility
947   \add@inserts\affixside@note%
948   \l@dlsn@te %space kept for backward compatibility
949   {\ledllfill\hb@xt@ \wd\one@line{\do@insidelineLhook\inserthangingsymbolL\new@lineL\l@
950     \l@drsn@te}}
951
```

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

```

952 \def\print@eledsectionL{%
953   \addtocounter{pstartL}{-1}%
954   \ifdefstring{\@eledsectnotoc}{L}{\ledsectnotoc}{%
955     \ifdefstring{\@eledsectmark}{L}{\ledsectnomark}%
956     \numdef{\temp@}{\l@dpscL-1}%
957     \xifinlist{\temp@}{\eled@sections@@}{\@nobreaktrue}{\@nobreakfalse}%
958     \@eled@sectioningtrue%
959     \csuse{eled@sectioning@the\l@dpscL}%
960     \@eled@sectioningfalse%
961     \global\csundef{eled@sectioning@the\l@dpscL}%
962     \if@RTL%
963       \hspace{-3\paperwidth}%
964       {\hbox{\l@dunhbox@line{\one@line}} \new@line}%
965     \else%
966       \hspace{3\paperwidth}%
967       {\new@line \hbox{\l@dunhbox@line{\one@line}}}%
968     \fi%
969     \vskip\eledsection@correcting@skip%
970 }
971

```

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

```

\doinlinedlineLhook 972 \newcommand*\dolineLhook[1]{\gdef\do@lineLhook{#1}}%
\doinlinedlineRhook 973 \newcommand*\dolineRhook[1]{\gdef\do@lineRhook{#1}}%
974 \newcommand*\doinlinedlineLhook[1]{\gdef\do@insidelineLhook{#1}}%
975 \newcommand*\doinlinedlineRhook[1]{\gdef\do@insidelineRhook{#1}}%
976

```

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

```

\dolineRhook 977 \newcommand*\do@lineLhook{}
\doinlinedlineLhook 978 \newcommand*\do@lineRhook{}
\doinlinedlineRhook 979 \newcommand*\do@insidelineLhook{}
980 \newcommand*\do@insidelineRhook{}
981

```

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

```

982 \newcommand*\do@lineR{%
983   \ledRcol@true%
984   \advance\countRline \@ne
985   \ifvbox\namebox{\l@dRcolrawbox\the\l@dpscR}%
986   {\vbadness=10000
987     \splittopskip=\z@
988     \do@lineRhook
989     \l@demtyd@ta
990     \global\setbox\one@lineR=\vsplit\namebox{\l@dRcolrawbox\the\l@dpscR}
991     to\baselineskip}%

```

```

992 \IfStrEq{\splitfirstmarks\parledgroup@}{begin}{\parledgroup@notes@startR}{-}
993 \unvbox\one@lineR \global\setbox\one@lineR=\lastbox
994 \getline@numR
995 \ifnum\@lockR>\@ne%
996   \inserthangingsymbolRtrue
997 \else%
998   \inserthangingsymbolRfalse%
999 \fi%
1000 \setbox\l@drighthbox
1001 \hb@xt@ \Rcolwidth{%
1002   \affixline@numR%
1003   \xifinlist{\the\l@dpscR}{\eled@sectionsR@@}%
1004   {\add@insertsR\affixside@noteR}%
1005   {\print@lineR}%
1006 }%
1007 \add@penaltiesR
1008 \global\advance\@donereallinesR\@ne
1009 \global\advance\@donetotallinesR\@ne
1010 \else
1011   \setbox\l@drighthbox \hb@xt@ \Rcolwidth{\hspace*{\Rcolwidth}}
1012   \global\advance\@donetotallinesR\@ne
1013 \fi
1014 \ledRcol@false%
1015 }
1016
1017

```

```

\print@lineR
\print@eledsectionR

```

17.3 Line and page number computation

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

```

1018 \newcommand*{\getline@numR}{%
1019   \global\advance\absline@numR \@ne
1020   \do@actionsR
1021   \do@ballastR
1022   \ifledgroupnotesR\else\ifnumberline
1023     \ifsublines@
1024       \ifnum\sub@lockR<\tw@
1025         \global\advance\subline@numR \@ne
1026       \fi
1027     \else
1028       \ifnum\@lockR<\tw@
1029         \global\advance\line@numR \@ne
1030         \global\subline@numR \z@
1031       \fi
1032     \fi
1033 \fi

```

```

1034 \fi
1035 }
1036 \newcommand*{\getline@numL}{%
1037   \global\advance\absline@num \@ne
1038   \do@actions
1039   \do@ballast
1040 \ifledgroupnotesL@\else\ifnumberline
1041   \ifsublines@
1042     \ifnum\sub@lock<\tw@
1043       \global\advance\subline@num \@ne
1044     \fi
1045   \else
1046     \ifnum\@lock<\tw@
1047       \global\advance\line@num \@ne
1048       \global\subline@num \z@
1049     \fi
1050   \fi
1051 \fi
1052 \fi
1053 }
1054
1055

```

`\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.

```

1056 \newcommand*{\do@ballastR}{\global\ballast@count=\z@
1057   \begingroup
1058   \advance\absline@numR \@ne
1059   \ifnum\next@actionlineR=\absline@numR
1060     \ifnum\next@actionR>-1001
1061       \global\advance\ballast@count by -\c@ballast
1062     \fi
1063   \fi
1064 \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 `\do@actions@fixedcodeR` line.
`\do@actions@nextR`

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

```

1065 \newcommand*{\do@actions@fixedcodeR}{%
1066   \ifcase\@l@dttempcnta%
1067   \or% % 1001
1068     \global\sublines@true
1069   \or% % 1002
1070     \global\sublines@false
1071   \or% % 1003
1072     \global\@lockR=\@ne
1073   \or% % 1004

```

```

1074 \ifnum\@lockR=\tw@
1075 \global\@lockR=\thr@@
1076 \else
1077 \global\@lockR=\z@
1078 \fi
1079 \or% % 1005
1080 \global\sub@lockR=\@ne
1081 \or% % 1006
1082 \ifnum\sub@lockR=\tw@
1083 \global\sub@lockR=\thr@@
1084 \else
1085 \global\sub@lockR=\z@
1086 \fi
1087 \or% % 1007
1088 \l@dskipnumbertrue
1089 \else
1090 \led@warn@BadAction
1091 \fi}
1092
1093
1094 \newcommand*{\do@actionsR}{%
1095 \global\let\do@actions@nextR=\relax
1096 \@l@dtmpcntb=\absline@numR
1097 \ifnum\@l@dtmpcntb<\next@actionlineR\else
1098 \ifnum\next@actionR>-1001\relax
1099 \global\page@numR=\next@actionR
1100 \ifbypage@R
1101 \global\line@numR \z@ \global\subline@numR \z@
1102 \fi
1103 \else
1104 \ifnum\next@actionR<-4999\relax % 9/05 added relax here
1105 \@l@dtmpcnta=-\next@actionR
1106 \advance\@l@dtmpcnta by -5001\relax
1107 \ifsublines@
1108 \global\subline@numR=\@l@dtmpcnta
1109 \else
1110 \global\line@numR=\@l@dtmpcnta
1111 \fi
1112 \else
1113 \@l@dtmpcnta=-\next@actionR
1114 \advance\@l@dtmpcnta by -1000\relax
1115 \do@actions@fixedcodeR
1116 \fi
1117 \fi
1118 \ifx\actionlines@listR\empty
1119 \gdef\next@actionlineR{1000000}%
1120 \else
1121 \glp\actionlines@listR\to\next@actionlineR
1122 \glp\actions@listR\to\next@actionR
1123 \global\let\do@actions@nextR=\do@actionsR

```

```

1124 \fi
1125 \fi
1126 \do@actions@nextR}
1127

```

17.4 Line number printing

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

```

\ch@cksub@l@ckR 1128
\ch@ck@l@ckR 1129 \providecommand*\l@dcalcnun}[3]{%
\fx@l@cksR 1130 \ifnum #1 > #2\relax
\affixline@numR 1131 \@l@tempcnta = #1\relax
1132 \advance\@l@tempcnta by -#2\relax
1133 \divide\@l@tempcnta by #3\relax
1134 \multiply\@l@tempcnta by #3\relax
1135 \advance\@l@tempcnta by #2\relax
1136 \else
1137 \@l@tempcnta=#2\relax
1138 \fi}
1139
1140 \newcommand*\ch@cksub@l@ckR}{%
1141 \ifcase\sub@lockR
1142 \or
1143 \ifnum\sublock@disp=\@ne
1144 \@l@tempcntb \z@ \@l@tempcnta \@ne
1145 \fi
1146 \or
1147 \ifnum\sublock@disp=\tw@
1148 \else
1149 \@l@tempcntb \z@ \@l@tempcnta \@ne
1150 \fi
1151 \or
1152 \ifnum\sublock@disp=\z@
1153 \@l@tempcntb \z@ \@l@tempcnta \@ne
1154 \fi
1155 \fi}
1156
1157 \newcommand*\ch@ck@l@ckR}{%
1158 \ifcase\@lockR
1159 \or
1160 \ifnum\lock@disp=\@ne
1161 \@l@tempcntb \z@ \@l@tempcnta \@ne
1162 \fi
1163 \or
1164 \ifnum\lock@disp=\tw@
1165 \else
1166 \@l@tempcntb \z@ \@l@tempcnta \@ne
1167 \fi
1168 \or

```

```

1169 \ifnum\lock@disp=\z@
1170 \l@dttempcntb \z@ \l@dttempcnta \@ne
1171 \fi
1172 \fi}
1173
1174 \newcommand*\f@x@l@cksR}{%
1175 \ifcase\@lockR
1176 \or
1177 \global\@lockR \tw@
1178 \or \or
1179 \global\@lockR \z@
1180 \fi
1181 \ifcase\sub@lockR
1182 \or
1183 \global\sub@lockR \tw@
1184 \or \or
1185 \global\sub@lockR \z@
1186 \fi}
1187
1188
1189 \newcommand*\affixline@numR}{%
1190 \ifledgroupnotesR@\else\ifnumberline
1191 \ifl@dskipnumber
1192 \global\l@dskipnumberfalse
1193 \else
1194 \ifsublines@
1195 \l@dttempcntb=\subline@numR
1196 \l@dcalcnnum{\subline@numR}{\c@firstsublinenumR}{\c@sublinenumincrementR}%
1197 \ch@cksub@lockR
1198 \else
1199 \l@dttempcntb=\line@numR
1200 \ifx\linenumberlist\empty
1201 \l@dcalcnnum{\line@numR}{\c@firstlinenumR}{\c@linenumincrementR}%
1202 \else
1203 \l@dttempcnta=\line@numR
1204 \edef\rem@inder{,\linenumberlist,\number\line@numR,%}
1205 \edef\sc@n@list{\def\noexpand\sc@n@list
1206 ###1,\number\l@dttempcnta,###2|{\def\noexpand\rem@inder{###2}}}%
1207 \sc@n@list\expandafter\sc@n@list\rem@inder|
1208 \ifx\rem@inder\empty\advance\l@dttempcnta\@ne\fi
1209 \fi
1210 \ch@ck@l@ckR
1211 \fi
1212 \ifnum\l@dttempcnta=\l@dttempcntb
1213 \if@twocolumn
1214 \if@firstcolumn
1215 \gdef\l@dld@ta{\llap{\leftlinenumR}}}%
1216 \else
1217 \gdef\l@drd@ta{\rlap{\rightlinenumR}}}%
1218 \fi

```



```

1219 \else
1220 \l@dttempcntb=\line@marginR
1221 \ifnum\l@dttempcntb>\@ne
1222 \advance\l@dttempcntb by\page@numR
1223 \fi
1224 \ifodd\l@dttempcntb
1225 \gdef\l@drd@ta{\rlap{{\rightlinenumR}}}%
1226 \else
1227 \gdef\l@dld@ta{\llap{{\leftlinenumR}}}%
1228 \fi
1229 \fi
1230 \fi
1231 \f@x@l@cksR
1232 \fi
1233 \fi
1234 \fi}

```

17.5 Pstart number printing in side

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

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

```

\affixpstart@numL
\affixpstart@numR 1235
\leftpstartnumR 1236 \newcommand*{\affixpstart@numL}{%
\rightpstartnumR 1237 \ifsidepstartnum
\leftpstartnumL 1238 \if@twocolumn
\rightpstartnumL 1239 \if@firstcolumn
\ifpstartnumR 1240 \gdef\l@dld@ta{\llap{{\leftpstartnumL}}}%
1241 \else
1242 \gdef\l@drd@ta{\rlap{{\rightpstartnumL}}}%
1243 \fi
1244 \else
1245 \l@dttempcntb=\line@margin
1246 \ifnum\l@dttempcntb>\@ne
1247 \advance\l@dttempcntb \page@num
1248 \fi
1249 \ifodd\l@dttempcntb
1250 \gdef\l@drd@ta{\rlap{{\rightpstartnumL}}}%
1251 \else
1252 \gdef\l@dld@ta{\llap{{\leftpstartnumL}}}%
1253 \fi
1254 \fi
1255 \fi
1256 }

```

```

1257 \newcommand*{\affixpstart@numR}{%
1258 \ifsidepstartnum
1259 \if@twocolumn
1260     \if@firstcolumn
1261         \gdef\l@dld@ta{\llap{\leftpstartnumR}}}%
1262     \else
1263         \gdef\l@drd@ta{\rlap{\rightpstartnumR}}}%
1264     \fi
1265 \else
1266     \l@dtempcntb=\line@marginR
1267     \ifnum\l@dtempcntb>\@ne
1268         \advance\l@dtempcntb \page@numR
1269     \fi
1270     \ifodd\l@dtempcntb
1271         \gdef\l@drd@ta{\rlap{\rightpstartnumR}}}%
1272     \else
1273         \gdef\l@dld@ta{\llap{\leftpstartnumR}}}%
1274     \fi
1275 \fi
1276 \fi
1277 }
1278
1279 \newcommand*{\leftpstartnumL}{
1280 \ifpstartnum
1281 \thepstartL
1282 \kern\linenumsep\global\pstartnumfalse\fi
1283 }
1284 \newcommand*{\rightpstartnumL}{
1285 \ifpstartnum\kern\linenumsep
1286 \thepstartL
1287 \global\pstartnumfalse\fi
1288 }
1289 \newif\ifpstartnumR
1290 \pstartnumRtrue
1291 \newcommand*{\leftpstartnumR}{
1292 \ifpstartnumR
1293 \thepstartR
1294 \kern\linenumsep\global\pstartnumRfalse\fi
1295 }
1296 \newcommand*{\rightpstartnumR}{
1297 \ifpstartnumR\kern\linenumsep
1298 \thepstartR
1299 \global\pstartnumRfalse\fi
1300 }

```

17.6 Add insertions to the vertical list

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

```

1301 \list@create{\inserts@listR}

\add@insertsR The right text version.
\add@inserts@nextR 1302 \newcommand*{\add@insertsR}{%
1303   \global\let\add@inserts@nextR=\relax
1304   \ifx\inserts@listR\empty \else
1305     \ifx\next@insertR\empty
1306       \ifx\insertlines@listR\empty
1307         \global\noteschanged@true
1308         \gdef\next@insertR{100000}%
1309       \else
1310         \gl@p\insertlines@listR\to\next@insertR
1311       \fi
1312     \fi
1313     \ifnum\next@insertR=\absline@numR
1314       \gl@p\inserts@listR\to\@insertR
1315       \@insertR
1316       \global\let\@insertR=\undefined
1317       \global\let\next@insertR=\empty
1318       \global\let\add@inserts@nextR=\add@insertsR
1319     \fi
1320   \fi
1321   \add@inserts@nextR}
1322

```

17.7 Penalties

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

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

```

\newcommand*{\add@penaltiesR}{\@l@tempcnta=\ballast@count
\ifnum\num@linesR>\@ne
\global\advance\par@lineR \@ne
\ifnum\par@lineR=\@ne
\advance\@l@tempcnta by \clubpenalty
\fi
\@l@tempcntb=\par@lineR \advance\@l@tempcntb \@ne
\ifnum\@l@tempcntb=\num@linesR
\advance\@l@tempcnta by \widowpenalty
\fi

```

```

\ifnum\par@lineR<\num@linesR
  \advance\@l@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.

```

1323 \newcommand*{\add@penaltiesL}{%
1324 \newcommand*{\add@penaltiesR}{%
1325

```

17.8 Printing leftover notes

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

```

1326 \newcommand*{\flush@notesR}{%
1327   \@xloop
1328   \ifx\inserts@listR\empty \else
1329     \glp\inserts@listR\to\@insertR
1330     \@insertR
1331     \global\let\@insertR=\undefined
1332   \repeat}
1333

```

18 Footnotes

18.1 Normal footnote formatting

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

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

Just a reminder of the arguments:

```

\printlinesR   #1      | #2 | #3 | #4 | #5 | #6 | #7
\printlinesR start-page | line | subline | end-page | line | subline | font
1334 \def\printlinesR#1|#2|#3|#4|#5|#6|#7|{\begingroup
1335   \setprintlines{#1}{#2}{#3}{#4}{#5}{#6}%
1336   \ifl@d@pnum #1\fullstop\fi
1337   \ifledplinenum \linenumr@p{#2}\Rlineflag\else \sympplinenum\fi
1338   \ifl@d@ssub \fullstop \sublinenumr@p{#3}\fi
1339   \ifl@d@dash \endashchar\fi
1340   \ifl@d@pnum #4\fullstop\fi
1341   \ifl@d@elin \linenumr@p{#5}\Rlineflag\fi
1342   \ifl@d@esl \ifl@d@elin \fullstop\fi \sublinenumr@p{#6}\fi
1343 \endgroup}
1344
1345 \let\ledsavedprintlines\printlines
1346

```

19 Cross referencing

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

```

1347 \list@create{\labelref@listR}
1348

```

`\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.

```

1349 \renewcommand*{\edlabel}[1]{%
1350   \ifl@d@pairing\ifautopar%
1351     \strut%
1352   \fi\fi%
1353   \@bsphack%
1354   \ifledRcol
1355     \write\linenum@outR{\string\@lab}%
1356     \ifx\labelref@listR\empty
1357       \xdef\label@refs{\zz@@@}%
1358     \else
1359       \gl@p\labelref@listR\to\label@refs
1360     \fi
1361     \ifvmode
1362       \advancelabel@refs
1363     \fi
1364     \protected@write\@auxout{%
1365       {\string\l@dmake@labelsR\space\thepage|\label@refs|\the\c@pstartR|{#1}}%
1366     \else
1367       \write\linenum@out{\string\@lab}%

```

```

1368 \ifx\labelref@list\empty
1369 \xdef\label@refs{\zz@@@}%
1370 \else
1371 \gl@p\labelref@list\to\label@refs
1372 \fi
1373 \ifvmode
1374 \advancelabel@refs
1375 \fi
1376 \protected@write\@auxout{}%
1377 {\string\l@dmake@labels\space\thepage|\label@refs|\the\c@pstart|{#1}}%
1378 \fi
1379 \@esphack}
1380
1381

```

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

```

1382 \def\l@dmake@labelsR#1|#2|#3|#4|#5{%
1383 \expandafter\ifx\csname the@label#5\endcsname \relax\else
1384 \led@warn@DuplicateLabel{#4}%
1385 \fi
1386 \expandafter\gdef\csname the@label#5\endcsname{#1|#2\Rlineflag|#3|#4}%
1387 \ignorespaces}
1388 \AtBeginDocument{%
1389 \def\l@dmake@labelsR#1|#2|#3|#4|#5{%
1390 }
1391

```

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

```

1392 \renewcommand*{\@lab}{%
1393 \ifledRcol
1394 \xright@appenditem{\linenumr@p{\line@numR}}|%
1395 \ifsublines@ \sublinenumr@p{\subline@numR}\else 0\fi}%
1396 \to\labelref@listR
1397 \else
1398 \xright@appenditem{\linenumr@p{\line@num}}|%
1399 \ifsublines@ \sublinenumr@p{\subline@num}\else 0\fi}%
1400 \to\labelref@list
1401 \fi}
1402

```

20 Side notes

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

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

```
\sidenotemargin* 1403 \WithSuffix\newcommand\sidenotemargin*[1]{%
1404   \l@dgetsidenote@margin{#1}
1405   \global\sidenote@marginR=\@l@tempcntb
1406   \global\sidenote@margin=\@l@tempcntb
1407 }
1408 \newcount\sidenote@marginR
1409 \global\sidenote@margin=\@ne
1410
```

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

```
1411 \newcommand*{\affixside@noteR}{%
1412   \def\sidenotecontent@{}%
1413   \numgdef{\itemcount@}{0}%
1414   \def\do##1{%
1415     \ifnumequal{\itemcount@}{0}%
1416     {%
1417       \appto\sidenotecontent@{##1}}% Not print not separator before the 1st note
1418     {\appto\sidenotecontent@{\sidenotesep ##1}}%
1419     }%
1420     \numgdef{\itemcount@}{\itemcount@+1}%
1421   }%
1422   \dolistloop{\l@dcnotesnotetext}%
1423   \ifnumgreater{\itemcount@}{1}{\led@err@ManySidenotes}{}%
1424   \gdef\@templ@d{%
1425     \gdef\@templ@n{\l@dcnotesnotetext\l@dcnotesnotetext@l\l@dcnotesnotetext@r}%
1426     \ifx\@templ@d\@templ@n \else%
1427       \if@twocolumn%
1428         \if@firstcolumn%
1429           \setl@dlp@rbox{##1}{\sidenotecontent@}%
1430         \else%
1431           \setl@drp@rbox{\sidenotecontent@}%
1432         \fi%
1433       \else%
1434         \@l@tempcntb=\sidenote@marginR%
1435         \ifnum\@l@tempcntb>\@ne%
1436           \advance\@l@tempcntb by\page@numR%
1437         \fi%
1438         \ifodd\@l@tempcntb%
1439           \setl@drp@rbox{\sidenotecontent@}%
1440           \gdef\sidenotecontent@{}%
1441           \numdef{\itemcount@}{0}%
1442           \dolistloop{\l@dcnotesnotetext@l}%
1443           \ifnumgreater{\itemcount@}{1}{\led@err@ManyLeftnotes}{}%
1444           \setl@dlp@rbox{\sidenotecontent@}%
1445         \else%
1446           \setl@dlp@rbox{\sidenotecontent@}%
1447           \gdef\sidenotecontent@{}%
1448           \numdef{\itemcount@}{0}%

```

```

1449      \dolistloop{\l@dcstotetext@r}%
1450      \ifnumgreater{\itemcount@}{1}{\led@err@ManyRightnotes}{}%
1451      \setl@drp@rbox{\sidenotecontent@}%
1452      \fi%
1453    \fi%
1454  \fi%
1455 }
1456

```

21 Familiar footnotes

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

```

1457 \renewcommand{\l@dbfnote}[1]{%
1458   \ifnumberedpar@
1459   \gdef\@tag{#1\relax}%
1460   \ifledRcol%
1461     \xright@appenditem{\noexpand\v1@dbfnote{\@expandonce\@tag}}{\@thefnmark}}%
1462     \to\inserts@listR
1463     \global\advance\insert@countR \@ne%
1464   \else%
1465     \xright@appenditem{\noexpand\v1@dbfnote{\@expandonce\@tag}}{\@thefnmark}}%
1466     \to\inserts@list
1467     \global\advance\insert@count \@ne%
1468   \fi
1469 \fi\ignorespaces}
1470

```

`\normalbfnoteX`

```

1471 \renewcommand{\normalbfnoteX}[2]{%
1472   \ifnumberedpar@
1473   \ifledRcol%
1474     \ifluatex
1475       \footnotelang@lua[R]%
1476     \fi
1477     \@ifundefined{xpg@main@language}%if polyglossia
1478     {}%
1479     {\footnotelang@poly[R]}%
1480     \protected@xdef\thisfootnote{\csuse{thefootnote#1}}%
1481     \xright@appenditem{\noexpand\vbfnoteX{#1}{#2}{\@expandonce\thisfootnote}}%
1482     \to\inserts@listR
1483     \global\advance\insert@countR \@ne%
1484   \else%
1485     \ifluatex
1486       \footnotelang@lua%
1487     \fi
1488     \@ifundefined{xpg@main@language}%if polyglossia
1489     {}%

```



```

1490      {\footnotelang@poly}%
1491      \protected@xdef\thisfootnote{\csuse{thefootnote#1}}%
1492      \xright@appenditem{\noexpand\vbfnoteX{#1}{#2}{\expandonce\thisfootnote}}%
1493                      \to\inserts@list
1494      \global\advance\insert@count \@ne%
1495      \fi
1496      \fi\ignorespaces}
1497

```

22 Verse

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

```

\inserthangingsymbolL
\inserthangingsymbolR 1498 \newif\ifinserthangingsymbolR
1499 \newcommand{\inserthangingsymbolL}{%
1500 \ifinserthangingsymbol%
1501     \ifinstanzaL%
1502         \hangingsymbol%
1503     \fi%
1504 \fi}
1505 \newcommand{\inserthangingsymbolR}{%
1506 \ifinserthangingsymbolR%
1507     \ifinstanzaR%
1508         \hangingsymbol%
1509     \fi%
1510 \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 1511 \newcommand{\correcthangingL}{%
1512 \ifl@dpadding\else%
1513     \ifinstanzaL%
1514         \ifinserthangingsymbol%
1515             \hskip \@ifundefined{sza@00}{0}{\expandafter%
1516                 \noexpand\csname sza@00\endcsname}\stanzaindentbase%
1517         \fi%
1518     \fi%
1519 \fi}
1520
1521 \newcommand{\correcthangingR}{%
1522 \ifl@dpadding\else%
1523     \ifinstanzaR%
1524         \ifinserthangingsymbolR%

```

```

1525      \hskip \ifundefined{sza@0@}{0}{\expandafter%
1526      \noexpand\csname sza@0@\endcsname}\stanzaindentbase%
1527      \fi%
1528    \fi%
1529 \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.

```

1530 \chardef\next=\catcode'\&
1531 \catcode'\&=\active
1532

```

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

```

1533 \newenvironment{astanza}{%
1534   \startstanzahook
1535   \catcode'\&\active
1536   \global\stanza@count\@ne\stanza@modulo\@ne
1537   \ifnum\usernamecount{sza@0@}=\z@
1538     \let\stanza@hang\relax
1539     \let\endlock\relax
1540   \else
1541   %%% \interlinepenalty\@M % this screws things up, but I don't know why
1542     \rightskip\z@ plus 1fil\relax
1543     \fi
1544     \ifnum\usernamecount{szp@0@}=\z@
1545       \let\sza@penalty\relax
1546     \fi
1547     \def&{%
1548       \endlock\mbox{}%
1549       \sza@penalty
1550       \global\advance\stanza@count\@ne
1551       \@astanza@line}%
1552     \def\&{%
1553       \endlock\mbox{}
1554       \pend
1555       \endstanzaextra}%
1556     \pstart
1557     \@astanza@line
1558   }{}
1559

```

\@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.

```

1560 \newcommand*{\@astanza@line}{%
1561   \ifnum\value{stanzaindentrepetition}=0
1562     \parindent=\csname sza@\number\stanza@count
1563     @\endcsname\stanzaindentbase

```

```

1564 \else
1565     \parindent=\csname sza@\number\stanza@modulo
1566             @\endcsname\stanzaindentbase
1567     \managestanza@modulo
1568 \fi
1569 \par
1570 \stanza@hang%\mbox{}%
1571 \ignorespaces}
1572

```

Lastly reset the modified category codes.

```

1573 \catcode'\&=\next
1574

```

23 Naming macros

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

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

```

\unhnamebox 1575 \providecommand*\newnamebox}[1]{%
\unvnamebox 1576 \expandafter\newbox\csname #1\endcsname}
\namebox 1577 \providecommand*\setnamebox}[1]{%
    1578 \expandafter\setbox\csname #1\endcsname}
    1579 \providecommand*\unhnamebox}[1]{%
    1580 \expandafter\unhbox\csname #1\endcsname}
    1581 \providecommand*\unvnamebox}[1]{%
    1582 \expandafter\unvbox\csname #1\endcsname}
    1583 \providecommand*\namebox}[1]{%
    1584 \csname #1\endcsname}
    1585

```

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

```

\usenamecount 1586 \providecommand*\newnamecount}[1]{%
    1587 \expandafter\newcount\csname #1\endcsname}
    1588 \providecommand*\usenamecount}[1]{%
    1589 \csname #1\endcsname}
    1590

```

24 Counts and boxes for parallel texts

In sequential text, each chunk (that enclosed by $\text{\pstart} \dots \text{\pend}$) 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.

```

1591 \newcount\l@dc@maxchunks
1592 \newcommand{\maxchunks}[1]{\l@dc@maxchunks=#1}
1593   \maxchunks{5120}
1594

```

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

```

1595 \newcount\l@dnumpsstartsR
1596

```

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

```

1597 \newcount\l@dpscl
1598 \newcount\l@dpscr
1599

```

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

```

1600 \newcommand*{\l@dsetuprawboxes}{%
1601   \@l@dttempcntb=\l@dc@maxchunks
1602   \loop\ifnum\@l@dttempcntb>\z@
1603     \newnamebox{\l@dLcolrawbox\the\@l@dttempcntb}
1604     \newnamebox{\l@dRcolrawbox\the\@l@dttempcntb}
1605     \advance\@l@dttempcntb \m@ne
1606   \repeat}
1607

```

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

```

1608 \newcommand*{\l@dsetupmaxlinecounts}{%
1609   \@l@dttempcntb=\l@dc@maxchunks
1610   \loop\ifnum\@l@dttempcntb>\z@
1611     \newnamecount{\l@dmaxlinesinpar\the\@l@dttempcntb}
1612     \advance\@l@dttempcntb \m@ne
1613   \repeat}
1614 \newcommand*{\l@dzeromaxlinecounts}{%
1615   \begingroup
1616   \@l@dttempcntb=\l@dc@maxchunks
1617   \loop\ifnum\@l@dttempcntb>\z@
1618     \global\usenamecount{\l@dmaxlinesinpar\the\@l@dttempcntb}=\z@
1619     \advance\@l@dttempcntb \m@ne
1620   \repeat
1621   \endgroup}
1622

```

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

```

1623 \AtBeginDocument{%
1624   \l@dsetuprawboxes
1625   \l@dsetupmaxlinecounts
1626   \l@dzeromaxlinecounts
1627   \l@dnumpsstartsL=\z@
1628   \l@dnumpsstartsR=\z@
1629   \l@dpscL=\z@
1630   \l@dpscR=\z@}
1631

```

25 Fixing babel

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

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

```

\ifl@dusedbabel A flag for checking if babel has been used as a package.
\l@dusedbabelfalse 1632 \newif\ifl@dusedbabel
\l@dusedbabeltrue 1633 \l@dusedbabelfalse

\ifl@dsamelang Suppress \ifl@dsamelang which didn't work and was not logical, because both
columns could have the same language but not the main language of the document.

\l@dchecklang

\l@dbbl@set@language In babel the macro \bbl@set@language{<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 \.

1634 \newcommand*{\l@dbbl@set@language}[1]{%
1635   \edef\languagename{#1}%
1636   \select@language{\languagename}%
1637   \if@filesw
1638     \protected@write\@auxout{}\string\select@language{\languagename}}%
1639   \addtocontents{toc}{\string\select@language{\languagename}}%
1640   \addtocontents{lof}{\string\select@language{\languagename}}%
1641   \addtocontents{lot}{\string\select@language{\languagename}}%
1642   \fi}
1643

```

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

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

```
\theledlanguageR 1644 \providecommand{\selectlanguage}[1]{%
1645 \newcommand*{\l@duselanguage}[1]{%
1646 \gdef\theledlanguageL{}
1647 \gdef\theledlanguageR{}
1648
```

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

```
1649 \AtBeginDocument{%
1650 \ifundefined{xpg@main@language}{%
1651 \ifundefined{bbl@main@language}{%
```

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

```
1652 \l@dusedbabelfalse
1653 \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.

```
1654 \l@dusedbabeltrue
1655 \let\l@doldselectlanguage\selectlanguage
1656 \let\l@doldbbl@set@language\bbl@set@language
1657 \let\bbl@set@language\l@dbbl@set@language
1658 \renewcommand{\selectlanguage}[1]{%
1659 \l@doldselectlanguage{#1}%
1660 \ifledRcol \gdef\theledlanguageR{#1}%
1661 \else \gdef\theledlanguageL{#1}%
1662 \fi}
```

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

```
1663 \renewcommand*{\l@duselanguage}[1]{%
1664 \l@doldselectlanguage{#1}}
```

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

```
1665 \gdef\theledlanguageL{\bbl@main@language}%
1666 \gdef\theledlanguageR{\bbl@main@language}%
1667 }%
1668 }
```

If on `Polyglossia`

```
1669 { \let\old@otherlanguage\otherlanguage%
1670 \renewcommand{\otherlanguage}[2][{}]{%
1671 \selectlanguage{#1}{#2}%
1672 \ifledRcol \gdef\theledlanguageR{#2}%
```

```

1673         \else           \gdef\theledlanguageL{#2}%
1674         \fi}%
1675         \let\l@duselanguage\select@language%
1676         \gdef\theledlanguageL{\xpg@main@language}%
1677         \gdef\theledlanguageR{\xpg@main@language}%
    That's it.
1678 }}

```

```

    \if@pstarts  \check@pstarts returns \@pstartstrue if there are any unprocessed chunks.
\@pstartstrue 1679 \newif\if@pstarts
\@pstartsfalse 1680 \newcommand*{\check@pstarts}{%
\check@pstarts 1681  \@pstartsfalse
                1682  \ifnum\l@dnumpstartsL>\l@dpscL
                1683  \@pstartstrue
                1684  \else
                1685  \ifnum\l@dnumpstartsR>\l@dpscR
                1686  \@pstartstrue
                1687  \fi
                1688  \fi
                1689 }
                1690

```

```

    \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 1691 \newif\ifaraw@text
                1692  \araw@textfalse
                1693 \newcommand*{\checkraw@text}{%
                1694  \araw@textfalse
                1695  \ifvbox\namebox{\l@dLcolrawbox\the\l@dpscL}
                1696  \araw@texttrue
                1697  \else
                1698  \ifvbox\namebox{\l@dRcolrawbox\the\l@dpscR}
                1699  \araw@texttrue
                1700  \fi
                1701  \fi
                1702 }
                1703

```

```

\@writelinesinparL  These write the number of text lines in a chunk to the section files, and then
\@writelinesinparR afterwards zero the counter.
                1704 \newcommand*{\@writelinesinparL}{%
                1705  \edef\next{%
                1706    \write\linenum@out{\string\@pend[\the\@donereallinesL]}}%
                1707  \next
                1708  \global\@donereallinesL \z@}
                1709 \newcommand*{\@writelinesinparR}{%
                1710  \edef\next{%
                1711    \write\linenum@outR{\string\@pendR[\the\@donereallinesR]}}%

```

```

1712 \next
1713 \global\@donereallinesR \z@}
1714

```

26 Parallel columns

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

```

\@eledsectionR 1715 \newsavebox{\@eledsectionL}%
1716 \newsavebox{\@eledsectionR}%

```

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

```

1717 \newcommand*{\Columns}{%
1718   \eledsection@correcting@skip=-\baselineskip% Correction for sections' titles
1719   \setcounter{pstartL}{\value{pstartLold}}
1720   \setcounter{pstartR}{\value{pstartRold}}
1721   \ifnum\l@dnumpstartsL=\l@dnumpstartsR\else
1722     \led@err@BadLeftRightPstarts{\the\l@dnumpstartsL}{\the\l@dnumpstartsR}%
1723   \fi

```

Start a group and zero counters, etc.

```

1724 \begingroup
1725   \l@dzeropenalties
1726   \endgraf\global\num@lines=\prevgraf
1727   \global\num@linesR=\prevgraf
1728   \global\par@line=\z@
1729   \global\par@lineR=\z@
1730   \global\l@dpscL=\z@
1731   \global\l@dpscR=\z@

```

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

```

1732   \check@pstarts
1733   \loop\if@pstarts
1734     \global\pstartnumtrue
1735     \global\pstartnumRtrue

```

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

```

1736   \global\advance\l@dpscL \@ne
1737   \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

```

1738   \checkraw@text
1739 {   \loop\ifaraw@text

```


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

```

1740      \l@duselanguage{\theledlanguageL}%
1741      \do@lineL
1742      \xifinlist{\the\l@dpscL}{\eled@sections@@}
1743      {%
1744      \ifdefstring{\@eledsectmark}{L}%
1745      {\csuse{eled@sectmark@the\l@dpscL}%
1746      }{}}%
1747      \global\csundef{eled@sectmark@the\l@dpscL}%
1748      \savebox{\@eledsectionL}{\parbox[t][t]{\Lcolwidth}{\vbox{\print@eledsectionL}}}%\vb
1749      }%
1750      {}%
1751      \l@duselanguage{\theledlanguageR}%
1752      \do@lineR
1753      \xifinlist{\the\l@dpscR}{\eled@sectionsR@@}
1754      {%
1755      \ifdefstring{\@eledsectmark}{R}%
1756      {\csuse{eled@sectmark@the\l@dpscR R}%
1757      }{}}%
1758      \global\csundef{eled@sectmark@the\l@dpscR R}%
1759      \savebox{\@eledsectionR}{\parbox[t][t]{\Rcolwidth}{\vbox{\print@eledsectionR}}}%\vb
1760      }%
1761      {}%
1762      \hb@xt@ \hsize{%
1763      \ifdefstring{\columns@position}{L}{\hfill }%
1764      \unhbox\l@leftbox%
1765      \ifhbox\@eledsectionL%
1766      \usebox{\@eledsectionL}%
1767      \fi%
1768      \print@columnseparator%
1769      \unhbox\l@rightbox%
1770      \ifhbox\@eledsectionR%
1771      \usebox{\@eledsectionR}%
1772      \fi%
1773      \ifdefstring{\columns@position}{R}{\hfill}%
1774      }%
1775      \checkraw@text
1776      \checkverseL
1777      \checkverseR
1778      \checkpb@columns
1779      \repeat}

```

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

```

1779      \@writelinesinparL
1780      \@writelinesinparR
1781      \check@pstarts
1782      \ifbypstart@
1783      \write\linenum@out{\string\@set[1]}

```

```

1784         \resetprevline@
1785     \fi
1786     \ifbypstart@R
1787         \write\linenum@outR{\string\@set[1]}
1788         \resetprevline@
1789     \fi
1790     \addtocounter{pstartL}{1}
1791     \addtocounter{pstartR}{1}
1792 \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.

```

1793     \flush@notes
1794     \flush@notesR
1795 \endgroup
1796 \global\l@dpscL=\z@
1797 \global\l@dpscR=\z@
1798 \global\l@dnpstartL=\z@
1799 \global\l@dnpstartR=\z@
1800 \ignorespaces
1801 \global\instanzaLfalse
1802 \global\instanzaRfalse}
1803

```

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

```

1804 \def\print@columnseparator{%
1805     \ifdim\beforecolumnseparator<0pt%
1806         \hfill%
1807     \else%
1808         \hspace{\beforecolumnseparator}%
1809     \fi%
1810     \columnseparator%
1811     \ifdim\aftercolumnseparator<0pt%
1812         \hfill%
1813     \else%
1814         \hspace{\aftercolumnseparator}%
1815     \fi%
1816 }%
1817 %\end{macrocode}
1818 % \end{macro}
1819 % \begin{macro}{\checkpb@columns}
1820 % \cs{checkpb@columns} prevent or make pagebreaking in columns, depending of the use of \
1821 %     \begin{macrocode}
1822
1823 \newcommand{\checkpb@columns}{%
1824     \newif\if@pb
1825     \newif\if@nopb

```

```

1826 \IfStrEq{\led@pb@setting}{before}{
1827 \numdef{\next@absline}{\the\absline@num+1}%
1828 \numdef{\next@abslineR}{\the\absline@numR+1}%
1829 \xifinlistcs{\next@absline}{l@prev@pb}{\@pbtrue}{}%
1830 \xifinlistcs{\next@abslineR}{l@prev@pbR}{\@pbtrue}{%
1831 \xifinlistcs{\next@absline}{l@prev@nopb}{\@nopbtrue}{}%
1832 \xifinlistcs{\next@abslineR}{l@prev@nopbR}{\@nopbtrue}{%
1833 }{}
1834 \IfStrEq{\led@pb@setting}{after}{
1835 \xifinlistcs{\the\absline@num}{l@prev@pb}{\@pbtrue}{}%
1836 \xifinlistcs{\the\absline@numR}{l@prev@pbR}{\@pbtrue}{%
1837 \xifinlistcs{\the\absline@num}{l@prev@nopb}{\@nopbtrue}{}%
1838 \xifinlistcs{\the\absline@numR}{l@prev@nopbR}{\@nopbtrue}{%
1839 }{}
1840 \if@nopb\nopagebreak[4]\enlargethispage{\baselineskip}\fi
1841 \if@pb\pagebreak[4]\fi
1842 }

```

`\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).

```

1843 \newcommand*{\columnseparator}{%
1844 \smash{\rule[-0.2\baselineskip]{\columnrulewidth}{1.05\baselineskip}}}
1845 \newdimen\columnrulewidth
1846 \columnrulewidth=\z@
1847

```

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

```

1848 \newcommand*{\columnspan}[1]{%
1849 \xdef\columns@position{#1}%
1850 }%
1851 \xdef\columns@position{R}%

```

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

```

1852 \newlength{\beforecolumnseparator}%
1853 \setlength{\beforecolumnseparator}{-2pt}%
1854
1855 \newlength{\aftercolumnseparator}%
1856 \setlength{\aftercolumnseparator}{-2pt}%
1857

```

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

`\setpositionliketwocolumns@L`

`\setwidthliketwocolumns@C`

`\setpositionliketwocolumns@C`

`\setwidthliketwocolumns@R`

`\setpositionliketwocolumns@R`

`\setwidthliketwocolumns@R`

in a **non- parallel** typesetting context to fix the position of the lines. The `\setnoteposition...` macros are called in `\xxxfootstart` in a **non- parallel** typesetting context to fix the position of notes block.

```

1858 \newcommand{\setwidthliketwocolumns@L}{%
1859 % Temporary dimension, initially equal to the standard hsize, i.e. text width
1860 %   \begin{macrocode}
1861   \newdimen\temp%
1862   \temp=\hsize%

      Hsize : Left + Right width
1863   \hsize=\Lcolwidth%
1864   \advance\hsize\Rcolwidth%

      Now, calculating the remaining space
1865   \advance\temp-\hsize%

      And multiply the hsize by 2/3 of this space
1866   \multiply\temp by 2%
1867   \divide\temp by 3%
1868   \advance\hsize\temp%
1869 }%
1870
1871 \newcommand{\setpositionliketwocolumns@L}{%
1872   \renewcommand{\ledrlfill}{\hfill}%
1873 }%
1874
1875 \newcommand{\setnotespositionliketwocolumns@L}{%
1876 }%
1877
1878
1879 \newcommand{\setwidthliketwocolumns@C}{%
1880 % Temporary dimension, initially equal to the standard hsize, i.e. text width
1881   \newdimen\temp%
1882   \temp=\hsize%
1883 % Hsize : Left + Right width
1884   \hsize=\Lcolwidth%
1885   \advance\hsize\Rcolwidth%
1886 % Now, calculating the remaining space
1887   \advance\temp-\hsize%

      And multiply the hsize by 1/2 of this space
1888   \divide\temp by 2%
1889   \advance\hsize\temp%
1890 }%
1891
1892 \newcommand{\setpositionliketwocolumns@C}{%
1893   \doinsidelinehook{\hfill}%
1894   \renewcommand{\ledrlfill}{\hfill}%
1895 }%
```

```

1896
1897 \newcommand{\setnotespositionliketwocolumns@C}{%
1898   \newdimen\temp%
1899   \newdimen\tempa%
1900   \temp=\hsize%
1901   \tempa=\Lcolwidth%
1902   \advance\tempa\Rcolwidth%
1903   \advance\temp-\tempa%
1904   \divide\temp by 2%
1905   \leftskip=\temp%
1906   \rightskip=-\temp%
1907 }%
1908
1909 \newcommand{\setwidthliketwocolumns@R}{%
    Temporary dimension, initially equal to the standard hsize, i.e. text width
1910   \newdimen\temp%
1911   \temp=\hsize%
    Hsize : Left + Right width
1912   \hsize=\Lcolwidth%
1913   \advance\hsize\Rcolwidth%
    Now, calculating the remaining space
1914   \advance\temp-\hsize%
    And multiply the hsize by 2/3 of this space
1915   \multiply\temp by 2%
1916   \divide\temp by 3%
1917   \advance\hsize\temp%
1918 }%
1919
1920 \newcommand{\setpositionliketwocolumns@R}{%
1921   \doinsidelinehook{\hfill}%
1922 }%
1923
1924 \newcommand{\setnotespositionliketwocolumns@R}{%
1925   \newdimen\temp%
1926   \newdimen\tempa%
1927   \temp=\hsize%
1928   \tempa=\Lcolwidth%
1929   \advance\tempa\Rcolwidth%
1930   \advance\temp-\tempa%
1931   \divide\temp by 2%
1932   \leftskip=\temp%
1933   \rightskip=-\temp%
1934 }%
1935

```

27 Parallel pages

This is considerably more complicated than parallel columns.

```

\l@minpagelinesL Counts for the number of lines on a left or right page, and the smaller of the
\l@minpagelinesR number of lines on a pair of facing pages.
1936 \newcount\l@minpagelinesL
1937 \newcount\l@minpagelinesR
1938 \newcount\l@minpagelines
1939

\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.

1940 \newcommand*\Pages{%
1941   \eledsection@correcting@skip=-2\baselineskip% line correcting for section titles.
1942   \setcounter{pstartL}{\value{pstartLold}}
1943   \setcounter{pstartR}{\value{pstartRold}}
1944   \parledgroup@notespacing@set@correction
1945   \typeout{}
1946   \typeout{***** PAGES *****}
1947   \ifnum\l@dnumstartsL=\l@dnumstartsR\else
1948     \led@err@BadLeftRightPstarts{\the\l@dnumstartsL}{\the\l@dnumstartsR}%
1949   \fi

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

1950   \cleartol@devenpage
1951   \begingroup
1952     \l@dzeropenalties
1953     \endgraf\global\l@num@lines=\prevgraf
1954     \global\l@num@linesR=\prevgraf
1955     \global\l@par@line=\z@
1956     \global\l@par@lineR=\z@
1957     \global\l@dpscL=\z@
1958     \global\l@dpscR=\z@
1959     \writtenlinesLfalse
1960     \writtenlinesRfalse

  Check if there are chunks to be processed.

1961     \check@pstarts
1962     \loop\if@pstarts

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

1963       \global\advance\l@dpscL \@ne
1964       \global\advance\l@dpscR \@ne

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

1965       \getlinesfromparlistL

```

```

1966      \getlinesfromparlistR
1967      \l@dcalc@maxoftwo{\@cs@linesinparL}{\@cs@linesinparR}%
1968      {\usernamecount{l@dmaxlinesinpar\the\l@dpscL}}%
1969      \check@pstarts
1970      \repeat

```

Zero the counts again, ready for the next bit.

```

1971      \global\l@dpscL=\z@
1972      \global\l@dpscR=\z@

```

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

```

1973      \getlinesfrompagelistL
1974      \getlinesfrompagelistR
1975      \l@dcalc@minoftwo{\@cs@linesonpageL}{\@cs@linesonpageR}%
1976      {\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.

```

1977      \check@pstarts
1978      \if@pstarts

```

Increment the chunk counts to get the first pair.

```

1979      \global\advance\l@dpscL \@ne
1980      \global\advance\l@dpscR \@ne

```

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

```

1981      \global\@donereallinesL=\z@
1982      \global\@donetotallinesL=\z@
1983      \global\@donereallinesR=\z@
1984      \global\@donetotallinesR=\z@

```

Start a loop over the boxes (chunks).

```

1985      \checkraw@text
1986 %      \begingroup
1987 {      \loop\ifraw@text

```

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

```

1988      \checkpageL
1989      \l@duselanguage{\theledlanguageL}%
1990 %%%      \begingroup
1991 {      \loop\ifl@dsamepage%

```

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

```

1992      \ifdefstring{\@eledsectnotoc}{L}{\ledsectnotoc}{\}
1993      \csuse{before@pstartL@the\l@dpscL}%
1994      \global\csundef{before@pstartL@the\l@dpscL}
1995      \do@lineL
1996      \xifinlist{\the\l@dpscL}{\eled@sections@@}

```

```

1997      {\print@eledsectionL}%
1998      {}%
1999      \advance\numpagelinesL \@ne
2000      \ifshiftedpstarts
2001          \ifdim\ht\l@dleftbox>0pt\hb@xt@ \hsize{\ledstrutL\unhbox\l@dleftbox}%
2002      \else%
2003          \parledgroup@correction@notespacing{L}
2004          \hb@xt@ \hsize{\ledstrutL\unhbox\l@dleftbox}%
2005      \fi

```

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

```

2006      \get@nextboxL%
2007      \ifprint@last@after@pendL%
2008          \csuse{after@pendL@the\l@dpscL}%
2009          \global\csundef{after@pendL@the\l@dpscL}%
2010      \fi%
2011      \checkpageL%
2012      \checkverseL
2013      \checkpbl
2014      \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.

```

2015      \ifl@dpagfull
2016          \@writelinesonpageL{\the\numpagelinesL}%
2017      \else
2018          \@writelinesonpageL{1000}%
2019      \fi

```

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

```

2020      \numpagelinesL \z@
2021      \parledgroup@correction@notespacing@init
2022      \clearl@dleftpage }%

```

Now do the same for the right text.

```

2023      \checkpageR
2024      \l@duselanguage{\theledlanguageR}%
2025 {
2026      \loop\ifl@dsamepage%
2027          \initnumbering@sectcountR
2028          \ifdefstring{\@eledsectnotoc}{R}{\ledsectnotoc}{%
2029              \csuse{before@pstartR@the\l@dpscR}%
2030              \global\csundef{before@pstartR@the\l@dpscR}
2031              \do@lineR
2032              \xifinlist{\the\l@dpscR}{\eled@sectionsR@@}

```



```

2032         {\print@eledsectionR}%
2033     }%
2034     \advance\numpagelinesR \@ne
2035     \ifshiftedpstarts
2036         \ifdim\ht\l@drightbox>0pt\hb@xt@ \hsize{\ledstrutR\unhbox\l@drightbox}\fi%
2037     \else%
2038         \parledgroup@correction@notespacing{R}
2039         \hb@xt@ \hsize{\ledstrutR\unhbox\l@drightbox}%
2040     \fi
2041     \get@nextboxR%
2042     \ifprint@last@after@pendR%
2043         \csuse{after@pendR@the\l@dpscR}%
2044         \global\csundef{after@pendR@the\l@dpscR}%
2045     \fi%
2046     \checkpageR%
2047     \checkverseR
2048     \checkpbR
2049     \repeat
2050     \ifl@dpagfull
2051         \@writelinesonpageR{\the\numpagelinesR}%
2052     \else
2053         \@writelinesonpageR{1000}%
2054     \fi
2055     \numpagelinesR=\z@
2056     \parledgroup@correction@notespacing@init

```

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

```

2057     \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.

```

2058     \checkraw@text
2059     \ifaraw@text
2060         \getlinesfrompagelistL
2061         \getlinesfrompagelistR
2062         \l@dcalcl@minoftwo{\@cs@linesonpageL}{\@cs@linesonpageR}%
2063         {\l@dminpagelines}%
2064     \fi
2065     \repeat}

```

We have now output the text from all the chunks.

```

2066     \fi

```

Make sure that there are no inserts hanging around.

```

2067     \flush@notes
2068     \flush@notesR
2069     \endgroup

```

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

```

2070 \global\l@dpscL=\z@
2071 \global\l@dpscR=\z@
2072 \global\l@dnumstartsL=\z@
2073 \global\l@dnumstartsR=\z@
2074 \global\instanzaLfalse
2075 \global\instanzaRfalse
2076 \ignorespaces}
2077

```

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

```

\ledstrutR 2078 \newcommand*{\ledstrutL}{\strut}
2079 \newcommand*{\ledstrutR}{\strut}
2080

```

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

```

2081 \providecommand{\cleartoevenpage}[1][\@empty]{%
2082 \clearpage
2083 \ifodd\c@page\hbox{#1}\clearpage\fi}
2084 \newcommand*{\cleartol@devenpage}{%
2085 \ifdim\pagetotal<\topskip% on an empty page
2086 \else
2087 \clearpage
2088 \fi
2089 \ifodd\c@page\hbox{ }\clearpage\fi}
2090 \newcommand*{\clearl@dleftpage}{%
2091 \clearpage
2092 \ifodd\c@page\else
2093 \led@err@LeftOnRightPage
2094 \hbox{ }%
2095 \cleardoublepage
2096 \fi}
2097 \newcommand*{\clearl@drighthouse}{%
2098 \clearpage
2099 \ifodd\c@page
2100 \led@err@RightOnLeftPage
2101 \hbox{ }%
2102 \cleartoevenpage
2103 \fi}
2104

```

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

```

\cs@linesinparR 2105 \newcommand*{\getlinesfromparlistL}{%
2106 \ifx\linesinpar@listL\empty

```

```

2107 \gdef\@cs@linesinparL{0}%
2108 \else
2109 \gl@p\linesinpar@listL\to\@cs@linesinparL
2110 \fi}
2111 \newcommand*\getlinesfromparlistR{%
2112 \ifx\linesinpar@listR\empty
2113 \gdef\@cs@linesinparR{0}%
2114 \else
2115 \gl@p\linesinpar@listR\to\@cs@linesinparR
2116 \fi}
2117

```

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

```

\cs@linesonpageR 2118 \newcommand*\getlinesfrompagelistL{%
2119 \ifx\linesonpage@listL\empty
2120 \gdef\@cs@linesonpageL{1000}%
2121 \else
2122 \gl@p\linesonpage@listL\to\@cs@linesonpageL
2123 \fi}
2124 \newcommand*\getlinesfrompagelistR{%
2125 \ifx\linesonpage@listR\empty
2126 \gdef\@cs@linesonpageR{1000}%
2127 \else
2128 \gl@p\linesonpage@listR\to\@cs@linesonpageR
2129 \fi}
2130

```

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

```

2131 \newcommand*\@writelinesonpageL[1]{%
2132 \edef\next{\write\linenum@out{\string\@lopL{#1}}}%
2133 \next}
2134 \newcommand*\@writelinesonpageR[1]{%
2135 \edef\next{\write\linenum@outR{\string\@lopR{#1}}}%
2136 \next}
2137

```

\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>.

```

2138 \newcommand*\l@dcalc@maxoftwo[3]{%
2139 \ifnum #2>#1\relax
2140 #3=#2\relax
2141 \else
2142 #3=#1\relax
2143 \fi}

```

```

2144 \newcommand*{\l@dcalc@minoftwo}[3]{%
2145   \ifnum #2<#1\relax
2146     #3=#2\relax
2147   \else
2148     #3=#1\relax
2149   \fi}
2150
\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@dpagetruefalse the maximum number of lines have been output then both \ifl@dpagetrue and
\l@dpagetruefalse \ifl@dsamepage are set FALSE.
\checkpageL
\checkpageR
2151 \newif\ifl@dsamepage
2152 \l@dsamepagetrue
2153 \newif\ifl@dpagetrue
2154
2155 \newcommand*{\checkpageL}{%
2156   \l@dpagetrue
2157   \l@dsamepagetrue
2158   \check@goal
2159   \ifdim\pagetotal<\ledthegoal
2160     \ifnum\l@dpagetrue<\l@dsamepagetrue
2161       \else
2162         \l@dsamepagefalse
2163         \l@dpagetruefalse
2164       \fi
2165     \else
2166       \l@dsamepagefalse
2167       \l@dpagetrue
2168     \fi%
2169   \ifprint@last@after@pendL%
2170     \l@dpagetruefalse%
2171     \l@dsamepagefalse%
2172     \print@last@after@pendLfalse%
2173   \fi%
2174 }%
2175
2176 \newcommand*{\checkpageR}{%
2177   \l@dpagetrue
2178   \l@dsamepagetrue
2179   \check@goal
2180   \ifdim\pagetotal<\ledthegoal
2181     \ifnum\l@dpagetrueR<\l@dsamepagetrue
2182       \else
2183         \l@dsamepagefalse
2184         \l@dpagetruefalse
2185       \fi

```

```

2186 \else
2187   \l@dsamepagefalse
2188   \l@dpagetrue
2189 \fi%
2190 \ifprint@last@after@pendR%
2191   \l@dpagetruefalse%
2192   \l@dsamepagefalse%
2193   \print@last@after@pendRfalse%
2194 \fi%
2195 }%
2196

```

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

```

2197 \newcommand{\checkpbL}{
2198   \IfStrEq{\led@pb@setting}{after}{
2199     \xifinlistcs{\the\absline@num}{l@prev@pb}{\l@dpagetrue\l@dsamepagefalse}{
2200       \xifinlistcs{\the\absline@num}{l@prev@nopb}{\l@dpagetruefalse\l@dsamepagetrue}{
2201     }}
2202   \IfStrEq{\led@pb@setting}{before}{
2203     \numdef{\next@absline}{\the\absline@num+1}
2204     \xifinlistcs{\next@absline}{l@prev@pb}{\l@dpagetrue\l@dsamepagefalse}{
2205       \xifinlistcs{\next@absline}{l@prev@nopb}{\l@dpagetruefalse\l@dsamepagetrue}{
2206     }}
2207 }
2208
2209 \newcommand{\checkpbR}{
2210   \IfStrEq{\led@pb@setting}{after}{
2211     \xifinlistcs{\the\absline@numR}{l@prev@pbR}{\l@dpagetrue\l@dsamepagefalse}{
2212       \xifinlistcs{\the\absline@numR}{l@prev@nopbR}{\l@dpagetruefalse\l@dsamepagetrue}{
2213     }}
2214   \IfStrEq{\led@pb@setting}{before}{
2215     \numdef{\next@abslineR}{\the\absline@numR+1}
2216     \xifinlistcs{\next@abslineR}{l@prev@pbR}{\l@dpagetrue\l@dsamepagefalse}{
2217       \xifinlistcs{\next@abslineR}{l@prev@nopbR}{\l@dpagetruefalse\l@dsamepagetrue}{
2218     }}
2219 }

```

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

```

2220 \newcommand{\checkverseL}{
2221   \ifinstanzaL
2222     \iflednopbinverse
2223       \ifinserthangingsymbol
2224         \numdef{\prev@abslineverse}{\the\absline@num-1}
2225         \IfStrEq{\led@pb@setting}{after}{\lednopbnum{\prev@abslineverse}}{
2226           \IfStrEq{\led@pb@setting}{before}{\ifnum\numpagelinesL<3\lednopbnum{\prev@abslineverse}\fi}{
2227         }

```

```

2228 \fi
2229 \fi
2230 }
2231 \newcommand{\checkverseR}{
2232 \ifinstanzaR
2233 \iflednopbinverse
2234 \ifinserthangingsymbolR
2235 \numgdef{\prev@abslineverse}{\the\absline@numR-1}
2236 \IfStrEq{\led@pb@setting}{after}{\lednopbnumR{\prev@abslineverse}}{}
2237 \IfStrEq{\led@pb@setting}{before}{\ifnum\numpagelinesR<3\ledpbnumR{\prev@abslineverse}}{}
2238 \fi
2239 \fi
2240 \fi
2241 }

```

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

```

2242 \newdimen\ledthegoal
2243 \ifshiftedpstarts
2244 \newcommand*{\goalfraction}{0.95}
2245 \else
2246 \newcommand*{\goalfraction}{0.9}
2247 \fi
2248
2249 \newcommand*{\check@goal}{%
2250 \ledthegoal=\goalfraction\pagegoal}
2251

```

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

```

\ifwrittenlinesL 2252 \newif\ifwrittenlinesL
2253 \newif\ifwrittenlinesR
2254

```

`\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.

```

2255 \newcommand*{\get@nextboxL}{%
2256 \ifvbox\namebox{1@dLcolrawbox\the\l@dpscl}% box is not empty

    The current box is not empty; do nothing.

2257 \else%                                box is empty

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

2258 \ifnum\usenamecount{1@dmaxlinesinpar\the\l@dpscl}>\@donetotallinesL
2259 \parledgroup@notes@endL
2260 \else

    Sufficient lines have been output.

2261 \ifnum\usenamecount{1@dmaxlinesinpar\the\l@dpscl}=\@donetotallinesL
2262 \parledgroup@notes@endL

```

```

2263     \fi
2264     \ifwrittenlinesL\else
    Write out the number of lines done, and set the boolean so this is only done once.
2265         \@writelinesinparL
2266         \writtenlinesLtrue
2267     \fi
2268     \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.
2269         \writtenlinesLfalse
2270         \ifbypstart@
2271             \ifnum\value{pstartL}<\value{pstartLold}
2272             \else
2273                 \global\line@num=0
2274                 \resetprevline@
2275             \fi
2276         \fi
2277 % Add the content of the optional argument of the previous \cs{pend}.
2278 %     \begin{macrocode}
2279         \csuse{after@pendL@the\l@dpscL}%
2280         \global\csundef{after@pendL@the\l@dpscL}%
2281 %     \end{macrocode}
2282 %     \begin{macrocode}
2283         \addtocounter{pstartL}{1}
2284         \global\pstartnumtrue
2285         \l@dcalcm@maxoftwo{\the\usernamecount{1@dmaxlinesinpar\the\l@dpscL}}%
2286             {\the\@donetotallinesL}%
2287             {\usernamecount{1@dmaxlinesinpar\the\l@dpscL}}%
2288         \global\@donetotallinesL \z@
2289         \global\advance\l@dpscL \@ne
    Add notes of parallel ledgroup.
2290         \parledgroup@notes@endL
2291         \parledgroup@correction@notespadding@final{L}
2292     \else
2293 % Add the content of the optional argument of the last \cs{pend}.
2294 %     \begin{macrocode}
2295         \print@last@after@pendLtrue%
2296 %     \end{macrocode}
2297 %     \begin{macrocode}
2298     \fi
2299 \fi
2300 \fi}

2301 \newcommand*{\get@nextboxR}{%
2302     \ifvbox\namebox{1@dRcolrawbox\the\l@dpscR}% box is not empty
2303     \else%
2304         \ifnum\usernamecount{1@dmaxlinesinpar\the\l@dpscR}>\@donetotallinesR

```

```

2305     \parledgroup@notes@endR
2306   \else
2307     \ifnum\usernamecount{l@dmxlinesinpar\the\l@dpscR}=\@donetotallinesR
2308       \parledgroup@notes@endR
2309     \fi
2310     \ifwrittenlinesR\else
2311       \@writelinesinparR
2312       \writtenlinesRtrue
2313     \fi
2314     \ifnum\l@dnumpstartsR>\l@dpscR
2315       \writtenlinesRfalse
2316       \ifbypstartR
2317         \ifnum\value{pstartR}<\value{pstartRold}
2318           \else
2319             \global\line@numR=0
2320             \resetprevline@
2321           \fi
2322         \fi
2323         \csuse{after@pendR@the\l@dpscR}%
2324         \global\csundef{after@pendR@the\l@dpscR}%
2325         \addtocounter{pstartR}{1}
2326         \global\pstartnumRtrue
2327         \l@dcalc@maxoftwo{\the\usernamecount{l@dmxlinesinpar\the\l@dpscR}}%
2328                           {\the\@donetotallinesR}%
2329                           {\usernamecount{l@dmxlinesinpar\the\l@dpscR}}%
2330         \global\@donetotallinesR \z@
2331         \global\advance\l@dpscR \@ne
2332         \parledgroup@notes@endR
2333         \parledgroup@correction@notes@spacing@final{R}
2334       \else
2335         \print@last@after@pendRtrue%
2336       \fi
2337     \fi
2338   \fi}
2339

```

28 Sections' titles' commands

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

```

2340 \newcommand{\eledsectnotoc}[1]{\xdef\@eledsectnotoc{#1}}
2341 \eledsectnotoc{R}

```

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

```

2342 \newcommand{\eledsectmark}[1]{\xdef\@eledsectmark{#1}}
2343 \eledsectmark{L}

```


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

2344 `\newskip\eledsection@correcting@skip`

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

2345 `\newwrite\eled@sectioningR@out`

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

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

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

2346 `\def\l@prev@pbR{}`

2347 `\def\l@prev@nopbR{}`

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

2348 `\newcommand{\ledpbR}{\write\linenum@outR{\string\led@pbR}}`

2349 `\newcommand{\ledpbnumR}[1]{\write\linenum@outR{\string\led@pbnumR{#1}}}`

2350 `\newcommand{\lednopbR}{\write\linenum@outR{\string\led@nopbR}}`

2351 `\newcommand{\lednopbnumR}[1]{\write\linenum@outR{\string\led@nopbnumR{#1}}}`

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

2352 `\newcommand{\led@pbR}{\listxadd{\l@prev@pbR}{\the\absline@numR}}`

2353 `\newcommand{\led@pbnumR}[1]{\listxadd{\l@prev@pbR}{#1}}`

2354 `\newcommand{\led@nopbR}{\listxadd{\l@prev@nopbR}{\the\absline@numR}}`

2355 `\newcommand{\led@nopbnumR}[1]{\listxadd{\l@prev@nopbR}{#1}}`

30 Parallel ledgroup

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

2356 `\newmarks\parledgroup@`

```

2357 \newmarks\parledgroup@series
2358 \newmarks\parledgroup@type

```

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

```

2359 \newcommand{\parledgroup@notes@startL}{%
2360   \ifnum\usenamecount{1@dmaxlinesinpar\the1@dpscL}>0%
2361     \IfStrEq{\splitfirstmarks\parledgroup@type}{footnoteX}{\csuse{bhooknoteX\splitfirstmarks}}
2362     \IfStrEq{\splitfirstmarks\parledgroup@type}{Xfootnote}{\csuse{bhookXnote\splitfirstmarks}}
2363   \fi%
2364   \global\ledgroupnotesL@true%
2365   \insert@noterule@ledgroup{L}%
2366 }
2367 \newcommand{\parledgroup@notes@startR}{%
2368   \ifnum\usenamecount{1@dmaxlinesinpar\the1@dpscR}>0%
2369     \IfStrEq{\splitfirstmarks\parledgroup@type}{footnoteX}{\csuse{bhooknoteX\splitfirstmarks}}
2370     \IfStrEq{\splitfirstmarks\parledgroup@type}{Xfootnote}{\csuse{bhookXnote\splitfirstmarks}}
2371   \fi%
2372   \global\ledgroupnotesR@true%
2373   \insert@noterule@ledgroup{R}%
2374 }

```

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

```

2375 \newcommand{\parledgroup@notes@endL}{%
2376   \global\ledgroupnotesL@false%
2377 }
2378 \newcommand{\parledgroup@notes@endR}{%
2379   \global\ledgroupnotesR@false%
2380 }

```

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

```

2381 \newcommand{\insert@noterule@ledgroup}[1]{
2382   \IfStrEq{\splitbotmarks\parledgroup@}{begin}{%
2383     \IfStrEq{\splitbotmarks\parledgroup@type}{Xfootnote}{
2384       \csuse{ifledgroupnotes#1@}
2385       \vskip\skip\csuse{mp\splitbotmarks\parledgroup@series footins}
2386       \csuse{\splitbotmarks\parledgroup@series footnoterule}
2387     \fi
2388   }
2389   {}
2390   \IfStrEq{\splitbotmarks\parledgroup@type}{footnoteX}{
2391     \csuse{ifledgroupnotes#1@}
2392     \vskip\skip\csuse{mpfootins\splitbotmarks\parledgroup@series}
2393     \csuse{footnoterule\splitbotmarks\parledgroup@series}
2394   \fi
2395   }{}

```

```

2396     }
2397   {}
2398 }

```

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

```
2399 \newcommand{\parledgroupnotespacing}{}

```

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

```

2400 \dimdef{\parledgroup@notespacing@correction}{0pt}
2401 \newcommand{\parledgroup@notespacing@set@correction}{%
2402   {\notefontsetup\parledgroupnotespacing\dimgdef{\temp@spacing}{\baselineskip}}%
2403   \dimgdef{\parledgroup@notespacing@correction}{\baselineskip-\temp@spacing}%
2404 }

```

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

```

2405 \newcommand{\parledgroup@correction@notespacing@init}{
2406   \dimdef{\parledgroup@notespacing@correction@accumulated}{0pt}
2407   \dimdef{\parledgroup@notespacing@correction@modulo}{0pt}
2408 }
2409 \parledgroup@correction@notespacing@init

```

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

```

2410 \newcommand{\parledgroup@correction@notespacing@final}[1]{
2411   \ifparledgroup
2412     \vspace{\parledgroup@notespacing@correction@accumulated}
2413     \parledgroup@correction@notespacing@init%
2414     \ifstrequal{#1}{L}{
2415       \numdef{\@checking}{\the\l@dpscL-1}
2416     }{
2417       \numdef{\@checking}{\the\l@dpscR-1}
2418     }
2419     \dimdef{\@beforenotes@current@diff}{\csuse{@parledgroup@beforenotes@\@checking L}-\csuse{@parledg
2420     \ifstrequal{#1}{L}%
2421       {% Left
2422         \ifdimgreater{\@beforenotes@current@diff}{0pt}{\vspace{-\@beforenotes@current@diff}}%
2423       }%
2424       {% Right
2425         \ifdimgreater{\@beforenotes@current@diff}{0pt}{\vspace{\@beforenotes@current@diff}}{}
2426       }%
2427     \fi
2428 }

```

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

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

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

```

2429 {}
2430 \newcommand{\parledgroup@correction@notespacing}[1]{%
2431     \csuse{ifledgroupnotes#1@}%
2432     \vspace{-\parledgroup@notespacing@correction}%
2433     \dimdef{\parledgroup@notespacing@correction@accumulated}{\parledgroup@notespacing@correction}%
2434     \dimdef{\parledgroup@notespacing@correction@modulo}{\parledgroup@notespacing@correction}%
2435     \ifdimless{\parledgroup@notespacing@correction@modulo}{\baselineskip}{\advance\num
2436     \dimdef{\parledgroup@notespacing@correction@modulo}{\parledgroup@notespacing@correction}%
2437     }% mean greater than equal
2438     \fi%
2439 }
```

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

```

2440 \dimdef\parledgroup@beforenotesL{0pt}
2441 \dimdef\parledgroup@beforenotesR{0pt}
```

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

```

2442 \newcommand{\parledgroup@beforenotes@save}[1]{
2443     \ifparledgroup
2444         \csdimgdef{@parledgroup@beforenotes@the\csuse{1@dnumstarts#1}#1}{\csuse{parledgroup@beforenotes@save#1}}
2445         \csdimgdef{parledgroup@beforenotes#1}{0pt}
2446     \fi
2447 }
```

31 The End

`i/codei`

Appendix A Some things to do when changing version

Appendix A.1 Migration to eledpar 1.4.3

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

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

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

See the two following examples:

With standard `\hangingsymbol`:

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

With the modification of `\hangingsymbol`:

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

References

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

Index

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

Symbols	<code>\@adv</code>	370, 630, 631
<code>\&</code>	1530, 1531, 1535, 1552, 1573	<code>\@afterindentfalse</code> 713
<code>\@M</code>	1541	<code>\@arabic</code> 222, 223, 762, 765

- \@astanza@line 1551, 1557, 1560
 - \@auxout 1364, 1376, 1638
 - \@beforenotes@current@diff
. 2419, 2422, 2425
 - \@chapter 714
 - \@checking 2415, 2417, 2419
 - \@cs@linesinparL 1967, 2105
 - \@cs@linesinparR 1967, 2105
 - \@cs@linesonpageL . . 1975, 2062, 2118
 - \@cs@linesonpageR . . 1975, 2062, 2118
 - \@currentlabel 805, 848
 - \@donereallinesL
. 907, 936, 1706, 1708, 1981
 - \@donereallinesR
. 907, 1008, 1711, 1713, 1983
 - \@donetotallinesL 907, 937,
940, 1982, 2258, 2261, 2286, 2288
 - \@donetotallinesR 907, 1009,
1012, 1984, 2304, 2307, 2328, 2330
 - \@eled@sectioningfalse 960
 - \@eled@sectioningtrue 958
 - \@eledsectionL . 1715, 1748, 1764, 1765
 - \@eledsectionR . 1715, 1759, 1769, 1770
 - \@eledsectmark . 955, 1744, 1755, 2342
 - \@eledsectnotoc 954, 1992, 2027, 2340
 - \@gobble 559
 - \@gobbletwo 564
 - \@insertR 1314–1316, 1329–1331
 - \@l@dttempcnta 443,
445, 447, 448, 452, 454, 456,
457, 1066, 1105, 1106, 1108,
1110, 1113, 1114, 1131–1135,
1137, 1144, 1149, 1153, 1161,
1166, 1170, 1203, 1206, 1208, 1212
 - \@l@dttempcntb 178, 180, 182,
1096, 1097, 1144, 1149, 1153,
1161, 1166, 1170, 1195, 1199,
1212, 1220–1222, 1224, 1245–
1247, 1249, 1266–1268, 1270,
1405, 1406, 1434–1436, 1438,
1601–1605, 1609–1612, 1616–1619
 - \@lab 563, 1355, 1367, 1392
 - \@lock 924, 1046
 - \@lockR 56, 314, 316, 318, 331,
477, 493, 494, 496, 497, 525,
526, 528, 995, 1028, 1072, 1074,
1075, 1077, 1158, 1175, 1177, 1179
 - \@lopL 588, 2132
 - \@lopR 559, 588, 2135
 - \@nl 292, 613, 615
 - \@nl@reg 341
 - \@nl@regR 292
 - \@nobreakfalse 771, 815, 957
 - \@nobreaktrue . . 769, 773, 813, 817, 957
 - \@noprtrue 1831, 1832, 1837, 1838
 - \@oldnobreak 769, 771, 813, 815, 866, 887
 - \@pbtrue 1829, 1830, 1835, 1836
 - \@pend 579, 1706
 - \@pendR 579, 1711
 - \@pstartfalse 1679
 - \@pstartstrue 1679
 - \@ref 549
 - \@ref@reg 577
 - \@schapter 714
 - \@set 402, 637, 638, 1783, 1787
 - \@startstanza 726, 727, 749, 750
 - \@sw 564
 - \@tag 1459, 1461, 1465
 - \@templ@d 1424, 1426
 - \@templ@n 1425, 1426
 - \@writelinesinparL . . 1704, 1779, 2265
 - \@writelinesinparR . . 1704, 1780, 2311
 - \@writelinesonpageL . 2016, 2018, 2131
 - \@writelinesonpageR . 2051, 2053, 2131
 - \@xloop 1327
- A**
- \absline@num
366, 436, 450, 469, 1037, 1827,
1835, 1837, 2199, 2200, 2203, 2224
 - \absline@numR 51, 239, 294,
297, 300, 433, 441, 462, 481,
515, 543, 554, 1019, 1058, 1059,
1096, 1313, 1828, 1836, 1838,
2211, 2212, 2215, 2235, 2352, 2354
 - \actionlines@list
. 284, 287, 436, 450, 469
 - \actionlines@listR
. 243, 260, 276, 279, 433,
441, 462, 481, 515, 543, 1118, 1121
 - \actions@list . 288, 437, 457, 471, 473
 - \actions@listR
. 243, 261, 280, 434, 448, 464,
466, 483, 492, 517, 524, 544, 1122
 - \add@inserts 933, 947
 - \add@inserts@nextR 1302
 - \add@insertsR 1004, 1302
 - \add@penaltiesL 935, 1323
 - \add@penaltiesR 1007, 1323
 - \addtocontents 1639–1641

- \addtocounter 868,
889, 953, 1790, 1791, 2283, 2325
- \advancelabel@refs 1362, 1374
- \advanceline 629, 660
- \affixline@num 931
- \affixline@numR 1002, 1128
- \affixpstart@numL 945, 1235
- \affixpstart@numR 1235
- \affixside@note 933, 947
- \affixside@noteR 1004, 1411
- \aftercolumnseparator . 4, 1811, 1852
- \appto 1417, 1418
- \araw@textfalse 1691
- \araw@texttrue 1691
- astanza (environment) 8, 1533
- \at@begin@pairs 690, 695, 696
- \at@every@pend 872, 893
- \at@every@pstart 808, 851
- \AtBeginDocument ... 1388, 1623, 1649
- \AtBeginPairs 4, 695

- B**
- \ballast@count 1056, 1061
- \bbl@main@language 1665, 1666
- \bbl@set@language 1656, 1657
- \beforecolumnseparator
..... 4, 1805, 1808, 1814, 1852
- \beginnumbering 7, 737, 776
- \beginnumberingR ... 42, 123, 737, 820
- \bfseries 762, 765
- \bypage@Rfalse 143, 158, 163
- \bypage@Rtrue 143, 153
- \bypstart@Rfalse 143, 154, 164
- \bypstart@Rtrue 143, 159

- C**
- \c@ballast 1061
- \c@chapter 104
- \c@chapterR 104
- \c@firstlinenumR 187, 1201
- \c@firstsublinenumR 191, 1196
- \c@linenumincrementR 187, 1201
- \c@page 613, 615, 2083, 2089, 2092, 2099
- \c@pstart 1377
- \c@pstartL 762
- \c@pstartR 765, 1365
- \c@section 105
- \c@sectionR 105
- \c@sublinenumincrementR .. 191, 1196
- \c@subsection 106
- \c@subsectionR 106
- \c@subsubsection 107
- \c@subsubsectionR 107
- \ch@ck@l@ckR 1128
- \ch@cksub@l@ckR 1128
- \ch@cksub@lockR 1197
- \chapter 699, 700, 709
- \chapterinpages 686, 700, 711
- \chardef 1530
- \check@goal 2158, 2179, 2242
- \check@pstarts
1679, 1732, 1781, 1961, 1969, 1977
- \checkpageL 1988, 2011, 2151
- \checkpageR 2023, 2046, 2151
- \checkpb@columns ... 1777, 1819, 1823
- \checkpbL 2013, 2197
- \checkpbR 2048, 2197
- \checkraw@text
..... 1691, 1738, 1774, 1985, 2058
- \checkverseL 1775, 2012, 2220
- \checkverseR 1776, 2047, 2220
- \cleardoublepage 2095
- \clearl@dleftpage 2022, 2081
- \clearl@drihtpage 2057, 2081
- \cleartoevenpage 2081
- \cleartol@devenpage 1950, 2081
- \closeout 95, 601, 608
- \columnrulewidth 4, 1843
- \Columns 3, 1717
- \columns@position .. 1762, 1772, 1848
- \columnseparator 4, 1810, 1843
- \columnspan 4, 1848
- \correcthangingL 949, 1511
- \correcthangingR 1511
- \countLline 902, 913
- \countRline 902, 984
- \critext 665
- \cs 1820, 2277, 2293
- \csdingdef 2444, 2445
- \csgdef 808,
809, 851, 852, 872, 873, 893, 894
- \csundef 961, 1747, 1758,
1994, 2009, 2029, 2044, 2280, 2324
- \csuse 959,
1480, 1491, 1745, 1756, 1993,
2008, 2028, 2043, 2279, 2323,
2361, 2362, 2369, 2370, 2384–
2386, 2391–2393, 2419, 2431, 2444

D

`\DeclareOption` 8–11
`\def@tempb` 161
`\dimdef` 2400, 2406, 2407,
 2419, 2433, 2434, 2436, 2440, 2441
`\dimen` 616, 617, 621–623, 627
`\dingdef` 2402, 2403
`\divide` 1133, 1867, 1888, 1904, 1916, 1931
`\do@actions` 1038
`\do@actions@fixedcodeR` 1065
`\do@actions@nextR` 1065
`\do@actionsR` 1020, 1065
`\do@ballast` 1039
`\do@ballastR` 1021, 1056
`\do@insidelineLhook` ... 949, 974, 977
`\do@insidelineRhook` 975, 977
`\do@lineL` 912, 1741, 1995
`\do@lineLhook` 917, 972, 977
`\do@lineR` 982, 1752, 2030
`\do@lineRhook` 973, 977, 988
`\do@lockoff` 512
`\do@lockoffL` 536
`\do@lockoffR` 512
`\do@lockon` 477
`\do@lockonL` 509
`\do@lockonR` 477
`\doinsidelinehook` 1893, 1921
`\doinsidelineLhook` 972
`\doinsidelineRhook` 972
`\dolineLhook` 972
`\dolineRhook` 972
`\dolistloop` 1422, 1442, 1449
`\dummy@ref` 558

E

`\edfont@info` 669, 672, 678, 681
`\edlabel` 1349
`\edtext` 665
`\eled@sectioningR@out` .. 70, 95, 2345
`\eled@sections@@` 932, 957, 1742, 1996
`\eled@sectionsR@@` 67, 1003, 1753, 2031
`\eledpar@error` 21, 30, 33, 35
`\eledsection@correcting@skip` ...
 969, 1718, 1941, 2344
`\eledsectmark` 12, 2342
`\eledsectnotoc` 12, 2340
`\empty` 80, 83, 276, 284,
 667, 676, 785, 829, 1118, 1200,
 1208, 1304–1306, 1317, 1328,
 1356, 1368, 2106, 2112, 2119, 2125

`\endashchar` 1339
`\endgraf` 862, 883, 1726, 1953
`\endline@num` 567, 573
`\endlock` 649, 1539, 1548, 1553
`\endnumbering` 7, 73, 127, 738
`\endnumberingR` 45, 73, 110, 122, 135, 738
`\endpage@num` 566, 573
`\endstanzaextra` 1555
`\endsub` 616
`\endsubline@num` 568, 574
`\enlargethispage` 1840
environments:
 astanza 8, 1533
 Leftside 5, 718
 pages 5, 686
 pairs 3, 686
 Rightside 5, 735
`\expandonce` 1461, 1465, 1481, 1492
`\extensionchars` 62, 116, 132, 140

F

`\f@x@l@cksR` 1128
`\first@linenum@out@Rfalse` .. 596, 602
`\first@linenum@out@Rtrue` 596
`\firstlinenum` 6, 196
`\firstlinenum*` 6, 196
`\firstsublinenum` 6, 196
`\firstsublinenum*` 6, 196
`\fix@page` 337, 344
`\flag@end` 616
`\flag@start` 616
`\flush@notes` 1793, 2067
`\flush@notesR` 1326, 1794, 2068
`\footnotelang@lua` 1475, 1486
`\footnotelang@poly` 1479, 1490
`\fullstop` . 235, 1336, 1338, 1340, 1342

G

`\get@linelistfile` 272
`\get@nextboxL` 2006, 2255
`\get@nextboxR` 2041, 2255
`\getline@numL` 923, 1036
`\getline@numR` 994, 1018
`\getlinesfrompagelistL`
 1973, 2060, 2118
`\getlinesfrompagelistR`
 1974, 2061, 2118
`\getlinesfromparlistL` ... 1965, 2105
`\getlinesfromparlistR` ... 1966, 2105

\glp .. 279, 280, 287, 288, 671, 680,
1121, 1122, 1310, 1314, 1329,
1359, 1371, 2109, 2115, 2122, 2128
\goalfraction 5, 2242

H

\hangingsymbol 10, 1502, 1508
\hb@xt@ 930, 939, 949, 1001,
1011, 1761, 2001, 2004, 2036, 2039
\hsize 803, 846, 1761, 1862–
1865, 1868, 1882, 1884, 1885,
1887, 1889, 1900, 1911–1914,
1917, 1927, 2001, 2004, 2036, 2039

I

\if@filesw 1637
\if@firstcolumn 1214, 1239, 1260, 1428
\if@nobreak 768, 812
\if@noeled@sec 68, 94
\if@nopb 1825, 1840
\if@pb 1824, 1841
\if@pstarts 1679, 1733, 1962, 1978
\if@RTL 962
\ifaraw@text ... 1691, 1739, 1987, 2059
\ifautopar 795, 839, 1350
\ifbypage@ 360
\ifbypage@R 143, 350, 1100
\ifbypstart@ 581, 1782, 2270
\ifbypstart@R ... 143, 585, 1786, 2316
\ifdefstring 954, 955,
1744, 1755, 1762, 1772, 1992, 2027
\ifdim ... 617, 621, 623, 627, 1805,
1811, 2001, 2036, 2085, 2159, 2180
\ifdimgreater 2422, 2425
\ifdimless 2435
\iffirst@linenum@out@R 596, 600
\ifhbox 1764, 1769
\ifinserthangingsymbol
..... 1500, 1514, 2223
\ifinserthangingsymbolR
..... 1498, 1506, 1524, 2234
\ifinstanzaL 716, 716, 1501, 1513, 2221
\ifinstanzaR 716, 717, 1507, 1523, 2232
\ifl@d@dash 1339
\ifl@d@elin 1341, 1342
\ifl@d@esl 1342
\ifl@d@pnum 1336, 1340
\ifl@d@ssub 1338
\ifl@dpagefull 2015, 2050, 2151
\ifl@dpadding 13, 1512, 1522

\ifl@dpairing 13, 77, 1350
\ifl@dsamelang 1634
\ifl@dsamepage 1991, 2025, 2151
\ifl@dskipnumber 1191
\ifl@dusedbabel 1632
\iflabelpstart 805, 848
\ifledgroupnotesL@ 1040
\ifledgroupnotesR@ 1022, 1190
\iflednopbinverse 2222, 2233
\ifledplinum 1337
\ifledRcol 13, 179, 201, 205, 209, 213,
257, 274, 338, 347, 372, 386,
403, 420, 432, 440, 461, 506,
533, 542, 551, 618, 624, 630,
637, 645, 650, 654, 659, 666,
1354, 1393, 1460, 1473, 1660, 1672
\ifluatex 1474, 1485
\ifnoteschanged@ 87
\ifnumberedpar@
... 778, 822, 858, 879, 1458, 1472
\ifnumbering 148, 774, 855
\ifnumberingR ... 43, 74, 112, 818, 876
\ifnumberline 1022, 1040, 1190
\ifnumberpstart ... 796, 840, 867, 888
\ifnumequal 1415
\ifnumgreater 1423, 1443, 1450
\ifodd 1224, 1249,
1270, 1438, 2083, 2089, 2092, 2099
\ifparledgroup 2411, 2443
\ifprint@last@after@pendL
..... 897, 2007, 2169
\ifprint@last@after@pendR
..... 897, 2042, 2190
\ifpst@rtedL 38, 782
\ifpst@rtedR 38, 826
\ifpstartnum 1280, 1285
\ifpstartnumR 1235
\ifshiftedpstarts 5, 2000, 2035, 2243
\ifsidepstartnum 797, 841, 1237, 1258
\ifstrempty 807, 850, 871, 892
\IfStrEq 921, 992, 1826,
1834, 2198, 2202, 2210, 2214,
2225, 2226, 2236, 2237, 2361,
2362, 2369, 2370, 2382, 2383, 2390
\ifstrequal 2414, 2420
\ifsublines@ 233,
326, 371, 404, 411, 442, 451,
463, 470, 482, 516, 572, 574,
1023, 1041, 1107, 1194, 1395, 1399
\ifvbox 914, 985, 1695, 1698, 2256, 2302

- \ifvmode 1361, 1373
 - \ifwidthliketwocolumns 11
 - \ifwritelinesL 2252, 2264
 - \ifwritelinesR 2253, 2310
 - \initnumbering@sectcmd 689, 703
 - \initnumbering@sectcountR
 - 66, 99, 119, 2026
 - \InputIfFileExists 69
 - \insert@count 548, 1467, 1494
 - \insert@countR 549, 1463, 1483
 - \insert@noterule@ledgroup
 - 2365, 2373, 2381
 - \inserthangingsymbolfalse 927
 - \inserthangingsymbolL 949, 1498
 - \inserthangingsymbolR 1498
 - \inserthangingsymbolRfalse 998
 - \inserthangingsymbolRtrue 996
 - \inserthangingsymboltrue 925
 - \insertlines@listR
 - 80, 243, 259, 554, 1306, 1310
 - \inserts@list 784, 1466, 1493
 - \inserts@listR 828, 1301,
 - 1304, 1314, 1328, 1329, 1462, 1482
 - \istanzaLfalse 1801, 2074
 - \istanzaLtrue 727
 - \istanzaRfalse 1802, 2075
 - \istanzaRtrue 750
 - \interlinepenalty 1541
 - \itemcount@ 1413, 1415,
 - 1420, 1423, 1441, 1443, 1448, 1450
- L**
- \l@d@nums 669, 672, 678, 681
 - \l@d@set 419, 645, 646
 - \l@dbbl@set@language 1634, 1657
 - \l@dbfnote 1457
 - \l@dc@maxchunks 790, 792,
 - 834, 836, 1591, 1601, 1609, 1616
 - \l@dcalc@maxoftwo
 - 1967, 2138, 2285, 2327
 - \l@dcalc@minoftwo .. 1975, 2062, 2138
 - \l@dcalcnun 1128
 - \l@dchecklang 1634
 - \l@dchset@num 293, 296, 419
 - \l@dcsnotetext 1422, 1425
 - \l@dcsnotetext@l 1425, 1442
 - \l@dcsnotetext@r 1425, 1449
 - \l@emptyd@ta 918, 989
 - \l@dend@stuff 63, 117, 133, 141
 - \l@dgetline@margin 177
 - \l@dgetsidenote@margin 1404
 - \l@dld@ta 946,
 - 1215, 1227, 1240, 1252, 1261, 1273
 - \l@dleftbox
 - .. 899, 929, 939, 1763, 2001, 2004
 - \l@dlinenumR 225
 - \l@dlsn@te 948
 - \l@dmake@labels 1377
 - \l@dmake@labelsR 1365, 1382
 - \l@dminpagelines
 - 1936, 1976, 2063, 2160, 2181
 - \l@dnumstartsl 789,
 - 790, 792, 794, 808, 809, 872,
 - 873, 1595, 1627, 1682, 1721,
 - 1722, 1798, 1947, 1948, 2072, 2268
 - \l@dnumstartsr 47, 833,
 - 834, 836, 838, 851, 852, 893,
 - 894, 1595, 1628, 1685, 1721,
 - 1722, 1799, 1947, 1948, 2073, 2314
 - \l@doldbbl@set@language 1656
 - \l@doldselectlanguage 1655, 1659, 1664
 - \l@dpagefullfalse
 - 2151, 2200, 2205, 2212, 2217
 - \l@dpagefulltrue
 - 2151, 2199, 2204, 2211, 2216
 - \l@dpagingfalse 14, 688, 708
 - \l@dpagingtrue 702
 - \l@dpairingfalse 13, 692, 707
 - \l@dpairingtrue 687, 701
 - \l@dpscL 914,
 - 919, 932, 956, 959, 961, 1597,
 - 1629, 1682, 1695, 1730, 1736,
 - 1742, 1745, 1747, 1796, 1957,
 - 1963, 1968, 1971, 1979, 1993,
 - 1994, 1996, 2008, 2009, 2070,
 - 2256, 2258, 2261, 2268, 2279,
 - 2280, 2285, 2287, 2289, 2360, 2415
 - \l@dpscR 985, 990, 1003,
 - 1598, 1630, 1685, 1698, 1731,
 - 1737, 1753, 1756, 1758, 1797,
 - 1958, 1964, 1972, 1980, 2028,
 - 2029, 2031, 2043, 2044, 2071,
 - 2302, 2304, 2307, 2314, 2323,
 - 2324, 2327, 2329, 2331, 2368, 2417
 - \l@drd@ta 949,
 - 1217, 1225, 1242, 1250, 1263, 1271
 - \l@drightbox
 - 899, 1000, 1011, 1768, 2036, 2039
 - \l@drsn@te 950

\l@dsamepagefalse	\led@mess@SectionContinued
.... <u>2151</u> , 2199, 2204, 2211, 2216 115, 131, 139
\l@dsamepagetrue	\led@nopbnumR
.... <u>2151</u> , 2200, 2205, 2212, 2217 2351, <u>2352</u>
\l@dsetupmaxlinecounts ..	\led@nopbR
<u>1608</u> , 1625 2350, <u>2352</u>
\l@dsetuprawboxes	\led@pb@setting
<u>1600</u> , 1624 1826, 1834, 2198, 2202,
\l@dskipnumberfalse	2210, 2214, 2225, 2226, 2236, 2237
1192	\led@pbnumR
\l@dskipnumbertrue 2349, <u>2352</u>
1088	\led@pbR
\l@dunhbox@line 2348, <u>2352</u>
949, 964, 967	\led@warn@BadAction
\l@dusedbabelfalse 1090
<u>1632</u> , 1652	\led@warn@BadAdvancelineLine ..
\l@dusedbabeltrue 389, 395
<u>1632</u> , 1654	\led@warn@BadAdvancelineSubline ..
\l@duselanguage 375, 381
.... <u>1644</u> , 1740, 1751, 1989, 2024	\led@warn@BadLineation
\l@dzeromaxlinecounts 166
<u>1608</u> , 1626	\led@warn@BadSetline
\l@dzeropenalties 861, 882, 1725, 1952 635
\l@prev@nopbR ...	\led@warn@BadSetlinenum
54, 2347, 2354, 2355 643
\l@prev@pbR	\led@warn@DuplicateLabel
53, 2346, 2352, 2353 1384
\l@pscL	\ledgroupnotesL@false
<u>1597</u> 2376
\l@pscR	\ledgroupnotesL@true
<u>1597</u> 2364
\label@refs	\ledgroupnotesR@false
1357, 1359, 1365, 1369, 1371, 1377 2379
\labelref@list	\ledgroupnotesR@true
1368, 1371, 1400 2372
\labelref@listR <u>1347</u> , 1356, 1359, 1396	\ledllfill
\language 949
1635, 1636, 1638–1641	\lednopb
\last@page@num 746
358, 364	\lednopbnum
\last@page@numR 2225, <u>2348</u>
<u>344</u>	\lednopbnumR
\lastbox 2236, <u>2348</u>
922, 993	\lednopbR
\lastskip 746, 2350
616, 622	\ledpb
\lcolwidth 745
.. 4, 5, <u>16</u> , 704, 803, 930, 939,	\ledpbnum
1748, 1863, 1884, 1901, 1912, 1928 2226
\led@err@BadLeftRightPstarts ...	\ledpbnumR
..... <u>29</u> , 1722, 1948 2237, <u>2348</u>
\led@err@LeftOnRightPage ...	\ledpbR
<u>32</u> , 2093 745, <u>2348</u>
\led@err@LineationInNumbered ...	\ledRcol@false
149 1014
\led@err@ManyLeftnotes	\ledRcol@true
1443 983
\led@err@ManyRightnotes	\ledRcolfalse
1450 15, 719, 752
\led@err@ManySidenotes	\ledRcoltrue
1423 736
\led@err@NumberingNotStarted ...	\ledrlfill
91 949, 1872, 1894
\led@err@numberingShouldHaveStarted	\ledsavedprintlines
..... 121 8, <u>1334</u>
\led@err@NumberingStarted	\ledsectnomark
44 955
\led@err@PendNoPstart	\ledsectnotoc
859, 880 954, 1992, 2027
\led@err@PendNotNumbered ...	\ledstrutL
856, 877 2001, 2004, 2078
\led@err@PstartInPstart ...	\ledstrutR
779, 823 2036, 2039, <u>2078</u>
\led@err@PstartNotNumbered .	\ledthegoal
775, 819 2159, 2180, <u>2242</u>
\led@err@RightOnLeftPage ...	\leftlinenumR
<u>32</u> , 2100 <u>225</u> , 1215, 1227
\led@err@TooManyPstarts	\leftpstartnumL
..... 24, 26, 791, 835 <u>1235</u>
\led@mess@NotesChanged	\leftpstartnumR
88 <u>1235</u>
	Leftside (environment)
 5, <u>718</u>
	\Leftsidehook
 725, <u>730</u>
	\Leftsidehookend
 729, <u>730</u>
	\line@list
 676, 680
	\line@list@stuff
 132
	\line@list@stuffR ..
 62, 116, 140, <u>598</u>

- \line@listR . 83, 243, 258, 574, 667, 671
 - \line@margin 182, 1245
 - \line@marginR 175, 1220, 1266
 - \line@num . 361, 393, 394, 396, 414,
425, 426, 454, 581, 1047, 1398, 2273
 - \line@numR 55,
232, 239, 298, 332, 351, 387,
388, 390, 407, 421, 422, 445,
567, 571, 585, 1029, 1101, 1110,
1199, 1201, 1203, 1204, 1394, 2319
 - \lineation 171, 172, 747
 - \lineation* 6, 171
 - \lineationR 6, 147, 173, 747
 - \linenum@out
. 613, 619, 625, 631, 638, 646,
651, 655, 1367, 1706, 1783, 2132
 - \linenum@outR 595, 601,
603, 608, 609, 615, 618, 624,
630, 637, 645, 650, 654, 659,
1355, 1711, 1787, 2135, 2348–2351
 - \linenumberlist 1200, 1204
 - \linenumincrement 6, 196
 - \linenumincrement* 6, 196
 - \linenummargin 175
 - \linenumr@p 1337, 1341, 1394, 1398
 - \linenumrepR 222, 232
 - \linenumsep
. 227, 229, 1282, 1285, 1294, 1297
 - \linesinpar@listL
. 250, 268, 582, 2106, 2109
 - \linesinpar@listR
. 250, 262, 586, 2112, 2115
 - \linesonpage@listL 269, 590, 2119, 2122
 - \linesonpage@listR 263, 593, 2125, 2128
 - \list@clear
. 258–265, 268, 269, 271, 784, 828
 - \list@clearing@reg 267
 - \list@create
. . . 243–248, 250–252, 1301, 1347
 - \listxadd 366, 2352–2355
 - \lock@disp 1160, 1164, 1169
 - \lock@off 503, 504, 512, 654, 655
 - \lock@on 650, 651
- M**
- \managestanza@modulo 1567
 - \maxchunks 3, 1591
 - \maxlinesinpar@list 250, 271
 - \memorydump 7, 724, 741
 - \memorydumpL 126, 724
 - \memorydumpR 126, 741
 - \message 61
 - \multiply 1134, 1866, 1915
- N**
- \n@num 540, 659
 - \n@num@reg 546
 - \namebox 914, 919, 985,
990, 1575, 1695, 1698, 2256, 2302
 - \NeedsTeXFormat 2
 - \new@line 964, 967
 - \new@lineL 612, 949
 - \new@lineR 614
 - \newbox 757, 899, 900, 1576
 - \newcommandx 767, 811, 854, 875
 - \newcounter 99–102, 187,
189, 191, 193, 760, 761, 763, 764
 - \newif 5, 39, 143,
144, 596, 716, 717, 897, 898,
1289, 1498, 1632, 1679, 1691,
1824, 1825, 2151, 2153, 2252, 2253
 - \newlength 1852, 1855
 - \newmarks 2356–2358
 - \newnamebox 1575, 1603, 1604
 - \newnamecount 1586, 1611
 - \newsavebox 1715, 1716
 - \newwrite 595, 2345
 - \next@absline
. . . . 1827, 1829, 1831, 2203–2205
 - \next@abslineR
. . . . 1828, 1830, 1832, 2215–2217
 - \next@action 288
 - \next@actionline 285, 287
 - \next@actionlineR
. 277, 279, 1059, 1097, 1119, 1121
 - \next@actionR 280, 1060,
1098, 1099, 1104, 1105, 1113, 1122
 - \next@insert 785
 - \next@insertR
829, 1305, 1308, 1310, 1313, 1317
 - \next@page@num 365, 437
 - \next@page@numR 59, 301, 303, 355, 434
 - \noindent 809, 852, 873, 894
 - \normal@page@break 366
 - \normal@page@breakR 52
 - \normal@pars 76, 788, 832
 - \normalbfnoteX 1471
 - \notefontsetup 2402
 - \noteschanged@true
. 81, 84, 668, 677, 1307

- `\notesXwidthliketwocolumns` 5
- `\num@lines` 862, 1726, 1953
- `\num@linesR` 756, 883, 1727, 1954
- `\numberedpar@true` 804, 847
- `\numberingRfalse` 75
- `\numberingRtrue` 49, 110, 136
- `\numberingtrue` 128
- `\numberpstartfalse` 8
- `\numberpstarttrue` 8
- `\numdef` 956, 1441, 1448,
1827, 1828, 2203, 2215, 2415, 2417
- `\numgdef` 1413, 1420, 2224, 2235
- `\numlabfont` 232
- `\numpagelinesL` 1936,
1999, 2016, 2020, 2160, 2226, 2435
- `\numpagelinesR`
1936, 2034, 2051, 2055, 2181, 2237
- O**
- `\old@otherlanguage` 1669
- `\old@startstanza` .. 726, 727, 749, 750
- `\oldchapter` 699, 709
- `\one@line` 919, 922, 949, 964, 967
- `\one@lineR` 756, 990, 993
- `\openout` 70, 603, 609
- `\otherlanguage` 1669, 1670
- P**
- `\p@pstartL` 806
- `\p@pstartR` 849
- `\PackageError` 21
- `\page@action` 302, 431, 560
- `\page@num` 283, 363, 1247
- `\page@numR` 254, 275, 353,
566, 571, 1099, 1222, 1268, 1436
- `\pagebreak` 1841
- `\pagegoal` 2250
- `\Pages` 5, 1940
- `pages (environment)` 5, 686
- `\pagetotal` 2085, 2159, 2180
- `pairs (environment)` 3, 686
- `\paperwidth` 963, 966
- `\par@line` 863, 1728, 1955
- `\par@lineR` 756, 884, 1729, 1956
- `\parbox` 1748, 1759
- `\parledgroup@` ... 921, 992, 2356, 2382
- `\parledgroup@beforenotes@save` ..
..... 870, 891, 2442
- `\parledgroup@beforenotesL` 2440
- `\parledgroup@beforenotesR` 2440
- `\parledgroup@correction@notespacing`
..... 2003, 2038, 2429
- `\parledgroup@correction@notespacing@final`
..... 2291, 2333, 2410
- `\parledgroup@correction@notespacing@init`
..... 2021, 2056, 2405, 2413
- `\parledgroup@notes@endL`
..... 2259, 2262, 2290, 2375
- `\parledgroup@notes@endR`
..... 2305, 2308, 2332, 2378
- `\parledgroup@notes@startL`
..... 921, 2359, 2375
- `\parledgroup@notes@startR`
..... 992, 2359, 2375
- `\parledgroup@notespacing@correction`
..... 2400, 2432–2434
- `\parledgroup@notespacing@correction@accumulated`
..... 2406, 2412, 2433
- `\parledgroup@notespacing@correction@modulo`
..... 2407, 2434–2436
- `\parledgroup@notespacing@set@correction`
..... 1944, 2400
- `\parledgroup@series`
..... 2357, 2361, 2362,
2369, 2370, 2385, 2386, 2392, 2393
- `\parledgroup@type` 2358,
2361, 2362, 2369, 2370, 2383, 2390
- `\parledgroupnotespacing` . 2399, 2402
- `\parledgroupseries@` 2356
- `\parledgrouptrue` 10
- `\parledgrouptype@` 2356
- `\pausenumbering` 739
- `\pausenumberingR` 109, 739
- `\pend` 6, 723, 744, 780, 1554
- `\pendL` 723, 854
- `\pendR` 744, 824, 875
- `\prev@abslineverse`
..... 2224–2226, 2235–2237
- `\prev@nopbR` 2346
- `\prev@pbR` 2346
- `\prevgraf`
..... 862, 883, 1726, 1727, 1953, 1954
- `\print@columnseparator` .. 1767, 1804
- `\print@eledsectionL`
..... 944, 952, 1748, 1997
- `\print@eledsectionR` . 1018, 1759, 2032
- `\print@last@after@pendLfalse` .. 2172
- `\print@last@after@pendLtrue` ... 2295
- `\print@last@after@pendRfalse` .. 2193
- `\print@last@after@pendRtrue` ... 2335

- \print@lineL 934, 944
- \print@lineR 1005, 1018
- \printlines 1345
- \printlinesR 8, 1334
- \ProcessOptions 12
- \protected@edef 805, 848
- \protected@write ... 1364, 1376, 1638
- \protected@xdef 1480, 1491
- \ProvidesPackage 3
- \pst@rtedLfalse 38
- \pst@rtedLtrue 129, 786
- \pst@rtedRfalse 40, 48, 78
- \pst@rtedRtrue 113, 137, 830
- \pstart 6, 27, 31, 721, 743, 1556
- \pstartL 721, 759
- \pstartnumfalse 1282, 1287
- \pstartnumRfalse 1294, 1299
- \pstartnumRtrue ... 1290, 1735, 2326
- \pstartnumtrue 1734, 2284
- \pstartR 743, 759

- R**
- \Rcolwidth
4, 5, 16, 705, 846, 1001, 1011,
1759, 1864, 1885, 1902, 1913, 1929
- \read@linelist 256, 599
- \rem@inder 1204, 1206–1208
- \resetprevline@ 1784, 1788, 2274, 2320
- \resumenumbering 740
- \resumenumberingR 109, 740
- \rightlinenumR 225, 1217, 1225
- \rightpstartnumL 1235
- \rightpstartnumR 1235
- Rightside (environment) 5, 735
- \Rightsidehook 730, 748
- \Rightsidehookend 730, 753
- \rlap 1217, 1225, 1242, 1250, 1263, 1271
- \Rlineflag 8, 220, 232, 1337, 1341, 1386
- \rule 1844

- S**
- \savebox 1748, 1759
- \sc@n@list 1205, 1207
- \secdef 714
- \section@num 130–132
- \section@numR 36,
50, 61, 62, 69, 70, 114–116, 138–140
- \select@language 1636, 1638–1641, 1675
- \selectlanguage 1644
- \set@line 665
- \set@line@action
..... 295, 400, 409, 416, 439, 562
- \setl@dlp@rbox 1429, 1444, 1446
- \setl@drp@rbox 1431, 1439, 1451
- \setline 633
- \setlinenum 641
- \setnamebox 794, 838, 1575
- \setnotepositionliketwocolumns@C
..... 1858
- \setnotepositionliketwocolumns@L
..... 1858
- \setnotepositionliketwocolumns@R
..... 1858
- \setnotespositionliketwocolumns@C
..... 1897
- \setnotespositionliketwocolumns@L
..... 1875
- \setnotespositionliketwocolumns@R
..... 1924
- \setpositionliketwocolumns@C ...
..... 1858, 1892
- \setpositionliketwocolumns@L ...
..... 1858, 1871
- \setpositionliketwocolumns@R ...
..... 1858, 1920
- \setprintlines 1335
- \setwidthliketwocolumns@C 1858, 1879
- \setwidthliketwocolumns@L 1858, 1858
- \setwidthliketwocolumns@R 1858, 1909
- \shiftedpstartsfalse 7
- \shiftedpstartstrue 6, 8, 9
- \shiftedversesfalse 7
- \shiftedversestrue 6
- \sidenote@margin 1406, 1409
- \sidenote@marginR 1403, 1434
- \sidenotecontent@
..... 1412, 1417, 1418, 1429, 1431,
1439, 1440, 1444, 1446, 1447, 1451
- \sidenotemargin 1403
- \sidenotemargin* 1403
- \sidenotesep 1418
- \skip 2385, 2392
- \skip@lockoff 504, 512
- \skipnumbering 9, 658
- \skipnumbering@reg 662
- \smash 1844
- \splitbotmarks 2382,
2383, 2385, 2386, 2390, 2392, 2393
- \splitfirstmarks
..... 921, 992, 2361, 2362, 2369, 2370

- \splittopskip 916, 987
 - \stanza@count 1536, 1550, 1562
 - \stanza@hang 1538, 1570
 - \stanza@modulo 1536, 1565
 - \stanzaindentbase
..... 1516, 1526, 1563, 1566
 - \startlock 649
 - \startstanzahook 1534
 - \startsub 616
 - \sub@action 311, 460, 561
 - \sub@change 60, 305, 306, 312
 - \sub@lock 1042
 - \sub@lockR 57, 320, 322, 324,
327, 478, 484, 485, 487, 488,
518, 519, 521, 1024, 1080, 1082,
1083, 1085, 1141, 1181, 1183, 1185
 - \sub@off 624, 625
 - \sub@on 618, 619
 - \subline@num 234, 361, 379,
380, 382, 412, 452, 1043, 1048, 1399
 - \subline@numR 235,
239, 328, 332, 351, 373, 374,
376, 405, 443, 568, 572, 1025,
1030, 1101, 1108, 1195, 1196, 1395
 - \sublinenumincrement 6, 196
 - \sublinenumincrement* 6, 196
 - \sublinenumr@p . 1338, 1342, 1395, 1399
 - \sublinenumrepR 222, 235
 - \sublines@false 58, 309, 1070
 - \sublines@true 307, 1068
 - \sublock@disp 1143, 1147, 1152
 - \sw@list@inedtextR 248, 265
 - \sw@listR 247, 264
 - \symplinenumber 1337
 - \sza@penalty 1545, 1549
- T**
- \temp 1861, 1862, 1865–1868,
1881, 1882, 1887–1889, 1898,
1900, 1903–1906, 1910, 1911,
1914–1917, 1925, 1927, 1930–1933
 - \temp@ 956, 957
 - \temp@spacing 2402, 2403
 - \tempa 1899, 1901–1903, 1926, 1928–1930
 - \textwidth 17, 19, 704, 705
 - \theledlanguageL ... 1644, 1740, 1989
 - \theledlanguageR ... 1644, 1751, 2024
 - \thepage 613, 615, 1365, 1377
 - \thepstart 722, 742
- U**
- \unhbox 1580,
1763, 1768, 2001, 2004, 2036, 2039
 - \unhnamebox 1575
 - \unvbox 922, 993, 1582
 - \unvnamebox 1575
 - \usebox 1765, 1770
 - \usenamecount 1537, 1544, 1586, 1618,
1968, 2258, 2261, 2285, 2287,
2304, 2307, 2327, 2329, 2360, 2368
- V**
- \value 783, 827, 1561,
1719, 1720, 1942, 1943, 2271, 2317
 - \vbadness 915, 986
 - \vbfnoteX 1481, 1492
 - \vbox 794, 838, 1748, 1759
 - \vl@dbfnote 1461, 1465
 - \vsplit 919, 990
- W**
- \wd 949
 - \widthliketwocolumns 4
 - \widthliketwocolumnstrue 11
 - \WithSuffix 171, 216–219, 1403
 - \writtenlinesLfalse 1959, 2269
 - \writtenlinesLtrue 2266
 - \writtenlinesRfalse 1960, 2315
 - \writtenlinesRtrue 2312
- X**
- \xifinlist 932,
957, 1003, 1742, 1753, 1996, 2031
 - \xifinlistcs 1829–
1832, 1835–1838, 2199, 2200,
2204, 2205, 2211, 2212, 2216, 2217
 - \Xnoteswidthliketwocolumns 5
 - \xpg@main@language 1676, 1677
 - \xright@appenditem
..... 433, 434, 436, 437,
441, 448, 450, 457, 462, 464,
466, 469, 471, 473, 481, 483,

492, 515, 517, 524, 543, 544,
 554, 570, 582, 586, 590, 593,
 1394, 1398, 1461, 1465, 1481, 1492 **Z**
 \zz@@@ 1357, 1369

Change History

v0.1.
 General: First public release 1

v0.2.
 General: Added section of babel re-
 lated code 59
 Fix babel problems 1
 \Columns: Added \l@dchecklang
 and \l@duselanguage to
 \Columns 63
 \Pages: Added \l@duselanguage
 to \Pages 70

v0.3.
 General: Added \do@lineLhook
 and \do@lineRhook 41
 Reorganize for ledarab 1
 \affixline@numR: Changed
 \affixline@numR to match new
 eledmac 45
 \do@actions@nextR: Used
 \do@actions@fixedcode in
 \do@actionsR 43
 \do@lineL: Added \do@lineLhook
 to \do@lineL 40
 Simplified \do@lineL by using
 macros for some common code 40
 \do@lineR: Changed \do@lineR
 similarly to \do@lineL 41
 Leftside: Added hooks into Left-
 side environment 34
 \flag@end: Removed extraneous
 spaces from \flag@end 31
 \ifledRcol: Moved \ifl@dpairing
 to eledmac 13
 \ifpst@rtedR: Moved \ifpst@rtedL
 to eledmac 14
 \l@dlinenumR: Simplified
 \leftlinenumR and \rightlinenumR
 by introducing \l@dlinenumR 20
 \l@dnumpstartsR: Moved
 \l@dnumpstartsL to eledmac . 58

\ledsavedprintlines: Simpli-
 fied \printlinesR by using
 \setprintlines 51
 \ledstrutR: Added \ledstrutL and
 \ledstrutR 72
 \normalbfnoteX: Removed
 extraneous spaces from
 \normalbfnoteX 54
 \Pages: Added \ledstrutL to
 \Pages 70
 Added \ledstrutR to \Pages . 71
 \Rightsidehookend: Added
 \Leftsidehook, \Leftsidehookend,
 \Rightsidehook and \Rightsidehookend
 34
 \sublinenumrepR: Added
 \linenumrepR and \sublinenumrepR
 20

v0.3.a.
 General: Minor \linenummargin
 fix 1
 \line@marginR: Don't just
 set \line@marginR in
 \linenummargin 18

v0.3.b.
 General: Improved parallel page
 balancing 1
 \Pages: Added \l@dminpagelines
 calculation for succeeding page
 pairs 71

v0.3.c.
 General: Compatibilty with Poly-
 glossia 1

v0.4.
 General: No more ledparpatch. All
 patches are now in the main
 file. 1

v0.5.
 General: Corrections about
 \section and other titles in

numbered sections	1	bering by pstart (like in eledmac 0.15).	40
v0.6.		Lineation can be by pstart (like in eledmac 0.15).	17
General: Be able to use <code>\chapter</code> in parallel pages.	1	New management of hangingsymbol insertion, preventing undesirable insertions.	55
v0.7.		Prevent shift of column separator when a verse is hanged	55
General: Option ‘shiftedverses’ which make there is no blank between two parallel verses with inequal length.	1	<code>\affixline@numR</code> : Changed <code>\affixline@numR</code> to allow to disable line numbering (like in eledmac 0.15).	45
v0.8.		<code>\Columns</code> : Line numbering by pstart.	64
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	<code>\get@nextboxR</code> : Change <code>\get@nextboxL</code> and <code>\get@nextboxR</code> to allow to disable line numbering (like in eledmac 0.15).	77
v0.9.		Pstart number can be printed in side	77
General: Possibility to number <code>\pstart</code>	8	v0.12.	
Possibility to number the pstart with the commands <code>\numberpstarttrue</code>	1	General: New new management of hangingsymbol insertion, preventing undesirable insertions.	55
<code>\iflledRcol</code> : Moved <code>\iflledRcol</code> and <code>\ifnumberingR</code> to eledmac	13	v1.0.	
v0.9.1.		General: Compatibility with eledmac. Change name to eledpar.	1
General: The numbering of the pstarts restarts on each <code>\beginnumbering</code>	1	Debug in lineation by pstart	17
v0.9.2.		v1.0.1.	
General: Debug : with <code>\Columns</code> , the hanging indentation now runs on the left columns and the hanging symbol is shown only when <code>\stanza</code> is used.	1	General: Correction on <code>\numberonlyfirstinline</code> with lineation by pstart or by page.	1
v0.9.3.		v1.1.	
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	General: Shiftedverses becomes shiftedpstarts.	1
v0.10.		<code>\pstartR</code> : Add <code>\labelpstarttrue</code> (from eledmac).	36
General: <code>\edlabel</code> commands on the right side are now correctly indicated.	1	v1.1.1.	
<code>\edlabel</code> commands which start a paragraph are now put in the right place.	1	<code>\pstartR</code> : Correct <code>\pstartR</code> bug introduced by 1.1.	36
v0.11.		v1.1.2.	
General: Change <code>\do@lineL</code> and <code>\do@lineR</code> to allow line num-		<code>\affixside@noteR</code> : Remove spurious space between line number and line content	53
		v1.2.	
		General: Support for <code>\led{section}</code> commands in parallel texts.	1

v1.2.1.	<code>\initnumbering@sectcountR</code> : For the right section, the counter is defined only once.	16	v1.5.0.	General: Add, as in <code>eledmac</code> , features to manage page breaks. . .	1
v1.3.	<code>\edtext</code> : Manage RTL language.	32	<code>\sublinenumincrement*</code> : Add starred version of <code>\firstlinenum</code> , <code>\linenumincrement</code> , <code>\firstsublinenum</code> , <code>\sublinenumincrement</code> to change both Left and Right-side.	19	
v1.3.1.	<code>\l@dbfnote</code> : Compatibility of standard footnotes with <code>eledmac</code> when theses footnotes contain any commands.	54	v1.6.0.	General: Add tool and documentation for parallel ledgroups . . .	10
v1.3.2.	General: Debug with some classes.	1	v1.7.0.	General: Add, as in <code>eledmac</code> , features to make crossrefs with <code>pstart</code> numbers.	1
v1.3.3.	General: Debugging the left notes of the right column.	53	v1.8.0.	General: <code>\beginnumbering</code> is defined only on <code>eledmac</code> , not on <code>eledpar</code>	15
	<code>\l@dbfnote</code> : Spurious space with footnote in right column.	54		<code>\l@dlsnote</code> , <code>\l@drsnote</code> and <code>\l@dcsnote</code> defined only one time, in <code>eledmac</code>	53
v1.3.4.	General: Allow use of commands in sidenotes, as introduced by <code>eledmac 1.0</code>	53		Add <code>\beforecolumnseparator</code> and <code>\aftercolumnseparator</code> . . .	4
v1.3.5.	<code>\normalbfnoteX</code> : Allows one to redefine <code>\thefootnoteX</code> with <code>alph</code> when some packages are loaded.	54		Add <code>\columnspan</code>	4
v1.4.	General: Added <code>\do@insidelineLhook</code> and <code>\do@insidelineRhook</code> . . .	41		Add, as in <code>eledmac</code> , new system of sectioning commands.	1
v1.4.1.	<code>\normalbfnoteX</code> : Fix bug with normal familiar footnotes when mixing RTL and LTR text. . .	54		Add, as in <code>eledmac</code> , option to insert something after <code>\pends</code> / verses.	1
	<code>astanza</code> : Enable the use of stanza indent repetition within stanza environment.	56		Add, as in <code>eledmac</code> , option to insert something between <code>\pstarts</code> / verse.	1
v1.4.3.	General: Corrects a false hanging verse when a verse is exactly the length of a line.	1		Change <code>\do@lineR</code> and <code>\do@lineL</code> to allow new sectioning commands.	40
	<code>\inserthangingsymbolR</code> : Hang verse is now not automatically flush right.	55		Compatibility with <code>musixtex</code> . . .	1
	<code>\pendL</code> : Spurious spaces in <code>\pendL</code>	38		Debug <code>eledmac</code> sectioning command after using <code>\resumenumbering</code>	1
	<code>\pendR</code> : Spurious spaces in <code>\pstartR</code>	39		New sectioning commands, as in <code>eledmac</code>	12
	<code>\pstartR</code> : Spurious spaces in <code>\pstartL</code> and <code>\pstartR</code>	36		<code>\Columns</code> : Modify <code>\Columns</code> to enable to add section's title. . . .	62
				Suppress <code>\l@dchecklang</code> from <code>\Columns</code>	63
				<code>\l@dchecklang</code> : Suppress <code>\l@dchecklang</code> which didn't	

work and was not logical, because both columns could have the same language but not the main language of the document.	59	<code>\doinsidelineLhook</code> and <code>\doinsidelineRhook</code>	41
<code>\Pages</code> : Modify <code>\Pages</code> to enable to add section's title.	68	<code>\Pages</code> : Debug blank pages when using optional argument in the last <code>\pend</code>	68
<code>\pendL</code> : As in <code>eledmac</code> , <code>\pendL</code> can have an optional argument.	38	<code>\resumenumberingR</code> : Debug <code>\resumenumberingR</code>	16
<code>\pendR</code> : As in <code>eledmac</code> , <code>\pendR</code> can have an optional argument.	39	v1.9.0.	
<code>\print@columnseparator</code> : Move some code of <code>\Columns</code> to <code>\print@columnseparator</code>	64	General: Add <code>\AtBeginPairs</code> macro.	4
<code>\pstartR</code> : As in <code>eledmac</code> , <code>\pendL</code> and <code>\pendR</code> can have an optional argument.	36	Compatibility with <code>\Xnoteswidthliketwocolumns</code> and <code>\notesXwidthliketwocolumns</code>	1
<code>\sidenotemargin*</code> : <code>\sidenotemargin</code> is now directly defined in <code>eledmac</code> to be able to manage <code>eledpar</code>	53	<code>\ifwidthliketwocolumns</code> : Added <code>widthliketwocolumns</code> option	13
Add <code>\sidenotemargin*</code>	53	<code>\theledlanguageR</code> : Debug left/right language switching with <code>polyglossia</code> . Don't write in <code>.aux</code> file when setting left/right lines.	61
<code>\theledlanguageR</code> : Correct left/right language setting with <code>polyglossia</code>	61	v1.9.1.	
v1.8.1.		<code>\ifledRcol</code> : Moved <code>\ifl@dpaging</code> to <code>eledmac</code>	13
<code>\do@lineL</code> : Fix a bug with critical notes at the beginning of a page, (maybe added by v1.8.0) (?).	40	v1.10.0.	
<code>\do@lineR</code> : Fix a bug with critical notes at the beginning of a page, added by v1.8.0 (?).	41	General: Compatibility with <code>\AtEveryPstart</code> and <code>\AtEveryPend</code>	1
v1.8.2.		Restore critical notes in <code>\eledsection</code> in parallel columns (this bug was added in 1.8.2).	1
General: Debug <code>\eledxxx</code> with some paper sizes	1	<code>\edlabel</code> : Prevent some bugs with cross-referencing when using <code>\autopar</code> in parallel texts.	51
Debug left and side note (bugs added by 1.8.0)	1	<code>\Pages</code> : Debug wrong pages splitting when no optional argument is used in last <code>\pend</code> (bug was added in v1.8.3).	68
<code>\eledpar@error</code> : Errors specific to <code>eledpar</code> send to <code>eledpar</code> handbook	14	Debug wrong parallel pages synchronization when an <code>\edtext</code> falls across two pages.	68
<code>\flag@end</code> : <code>\flag@start</code> and <code>\flag@end</code> are now defined only one time for <code>eledmac</code> and <code>eledpar</code>	31	v1.10.1.	
<code>\lineation*</code> : Add <code>\lineation*</code>	18	<code>\line@list@stuffR</code> : Revert modification of 1.4.2 which makes bug with numbering. Leave vertical mode to solve spurious space before <code>minipage</code>	30
v1.8.3.		v1.11.0.	
General: Add <code>\noeledxxx</code> , as in <code>eledmac</code>	1	General: Compatibility of standard footnotes with some <code>biblatex</code>	
<code>\doinsidelineRhook</code> : Added <code>\dolineLhook</code> , <code>\dolineRhook</code> ,			

styles.	1	now defined only in eledmac. .	32
<code>\edtext</code> : <code>\critext</code> and <code>\edtext</code> are			