

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 v1.9.0, last revised 2014/09/16.

[†]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	16
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	36
17 Paragraph decomposition and reassembly	38
17.1 Boxes, counters, <code>\pstart</code> and <code>\pend</code>	38
17.2 Processing one line	42
17.3 Line and page number computation	45
17.4 Line number printing	48
17.5 Pstart number printing in side	50
17.6 Add insertions to the vertical list	52
17.7 Penalties	52
17.8 Printing leftover notes	53
18 Footnotes	53
18.1 Normal footnote formatting	53
19 Cross referencing	54
20 Side notes	56

<i>List of Figures</i>	3
21 Familiar footnotes	57
22 Verse	58
23 Naming macros	60
24 Counts and boxes for parallel texts	61
25 Fixing babel	62
26 Parallel columns	65
27 Parallel pages	71
28 Sections' titles' commands	81
29 Page break/no page break, depending on the specific line	81
30 Parallel ledgroup	82
31 The End	85
Appendix A Some things to do when changing version	86
Appendix A.1 Migration to eledpar 1.4.3	86
References	86
Index	86
Change History	97

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}
...
\Columns
\end{pairs}
```

`\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
<code>\Rcolwidth</code>	columns, respectively. By default, these are:
	<code>\setlength{\Lcolwidth}{0.45\textwidth}</code>
	<code>\setlength{\Rcolwidth}{0.45\textwidth}</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.
<code>\columnseparator</code>	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>
	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:
<code>\aftercolumnseparator</code>	
	<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.
	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>
	If you want to come back to the previous behavior, just set them with
<code>\widthliketwocolumns</code>	a negative value. If you want to mix texts in columns and text with-
	out 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>\notesXwidthliketwocolumns</code>	<code>\Xnoteswidthliketwocolumns</code> and <code>\notesXwidthliketwocolumns</code> to align the
	critical/familiar footnotes with the two colums. <code>eledmac</code> handbook for more de-
	tails.

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

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

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

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

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

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

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

5 Left and right texts

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

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

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

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

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

number. Using these macros inside the **Leftside** and **Rightside** environments gives you independent control over the left and right numbering schemes. The `\firstsublinenum` and `\sublinenumincrement` macros correspondingly set the numbering scheme for sublines. The starred versions change both left and right numbering schemes.

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

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

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

`\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}
  ...
\end{pairs}
```

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

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

    \strut &
\end{astanza}
\end{pairs}
```

```

\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 taken from left side. You can change it, using `\eledsectmark{<arg>}`, where `<arg>` could be L (for left side) or R (for right side).

11 Implementation overview

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

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

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

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

12 Preliminaries

Announce the name and version of the package, which is targetted for LaTeX2_ε. The package also requires the `eledmac` package.

```

1 (*code)
2 \NeedsTeXFormat{LaTeX2e}
3 \ProvidesPackage{eledpar}[2014/09/16 v1.9.0 eledmac extension for parallel texts]%
4

```

With the option ‘shiftedpstarts’ a long pstart one 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. \ifl@dpairing is defined in eledmac.

```

13 \l@dpairingfalse
14 \newif\ifl@dpaging
15 \l@dpagingfalse
16 \ledRcolfalse

```

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

```

\Rcolwidth 17 \newdimen\Lcolwidth
18 \Lcolwidth=0.45\textwidth
19 \newdimen\Rcolwidth
20 \Rcolwidth=0.45\textwidth
21

```

12.1 Messages

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

```

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

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

\led@err@LeftOnRightPage
\led@err@RightOnLeftPage
33 \newcommand*{\led@err@LeftOnRightPage}{%
34 \eledpar@error{The left page has ended on a right page}\@ehc}}
35 \newcommand*{\led@err@RightOnLeftPage}{%
36 \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.

```

37 \newcount\section@numR
38 \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`.

```

39 \pst@rtedLfalse
40 \newif\ifpst@rtedR
41 \pst@rtedRfalse
42

```

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

```

43 \newcommand*{\beginnumberingR}{%
44 \ifnumberingR

```



```

45   \led@err@NumberingStarted
46   \endnumberingR
47   \fi
48   \global\l@dnumpstartsR \z@
49   \global\pst@rtedRfalse
50   \global\numberingRtrue
51   \global\advance\section@numR \@ne
52   \global\absline@numR \z@
53   \gdef\normal@page@breakR{}
54   \gdef\l@prev@pbR{}
55   \gdef\l@prev@nopbR{}
56   \global\line@numR \z@
57   \global\@lockR \z@
58   \global\sub@lockR \z@
59   \global\sublines@false
60   \global\let\next@page@numR\relax
61   \global\let\sub@change\relax
62   \message{Section \the\section@numR R }%
63   \line@list@stuffR{\jobname.\extensionchars\the\section@numR R}%
64   \l@dend@stuff
65   \setcounter{pstartR}{1}
66   \begingroup
67   \initnumbering@sectcountR
68   \gdef\eled@sectionsR@{ }%
69   \if@noeled@sec\else%
70     \makeatletter\inputIfFileExists{\jobname.eledsec\the\section@numR R}{ }{\makeatother%
71       \immediate\openout\eled@sectioningR@out=\jobname.eledsec\the\section@numR R\relax%
72     }%
73 }

```

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

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

```

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

```

```

88   \ifnoteschanged@
89   \led@mess@NotesChanged
90   \fi
91 \else
92   \led@err@NumberingNotStarted
93 \fi
94 \endgroup
95 \if@noeled@sec\else%
96   \immediate\closeout\eled@sectioningR@out%
97 \fi%
98 }
99

```

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

```

100 \newcounter{chapterR}
101 \newcounter{sectionR}
102 \newcounter{subsectionR}
103 \newcounter{subsubsectionR}
104 \newcommand{\initnumbering@sectcountR}{
105   \let\c@chapter\c@chapterR
106   \let\c@section\c@sectionR
107   \let\c@subsection\c@subsectionR
108   \let\c@subsubsection\c@subsubsectionR
109 }

```

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

```

110 \newcommand*{\pausenumberingR}{%
111   \endnumberingR\global\numberingRtrue}
112 \newcommand*{\resumenumberingR}{%
113   \ifnumberingR
114     \global\pst@rtedRtrue
115     \global\advance\section@numR \@ne
116     \led@mess@SectionContinued{\the\section@numR R}%
117     \line@list@stuffR{\jobname.\extensionchars\the\section@numR R}%
118     \l@dend@stuff
119     \begingroup%
120     \initnumbering@sectcountR%
121   \else
122     \led@err@numberingShouldHaveStarted
123     \endnumberingR
124     \beginnumberingR
125   \fi}
126

```

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

```

127 \newcommand*{\memorydumpL}{\%
128   \endnumbering
129   \numberingtrue
130   \global\pst@rtedLtrue
131   \global\advance\section@num \@ne
132   \led@mess@SectionContinued{\the\section@num}%
133   \line@list@stuff{\jobname.\extensionchars\the\section@num}%
134   \l@dend@stuff}
135 \newcommand*{\memorydumpR}{\%
136   \endnumberingR
137   \numberingRtrue
138   \global\pst@rtedRtrue
139   \global\advance\section@numR \@ne
140   \led@mess@SectionContinued{\the\section@numR R}%
141   \line@list@stuffR{\jobname.\extensionchars\the\section@numR R}%
142   \l@dend@stuff}
143

```

14 Line counting

14.1 Choosing the system of lineation

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

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

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

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

`\ifbypage@R` •

`\bypage@Rtrue` •

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

```

144 \newif\ifbypage@R
145 \newif\ifbypstart@R
146   \bypage@Rfalse
147   \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`.

```

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

```

```

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

```

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

```

172 \WithSuffix\newcommand\lineation*[1]{%
173     \lineation{#1}%
174     \lineationR{#1}%
175 }%

```

`\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.

```

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

```

By default put right text numbers at the right.

```

186 \line@marginR=\@ne
187

```

`\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.

```
188 \newcounter{firstlinenumR}
189 \setcounter{firstlinenumR}{5}
190 \newcounter{linenumincrementR}
191 \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.

```
192 \newcounter{firstsublinenumR}
193 \setcounter{firstsublinenumR}{5}
194 \newcounter{sublinenumincrementR}
195 \setcounter{sublinenumincrementR}{5}
196
```

`\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 `\providing` them. The starred
`\firstsublinenum` versions are specific to eledpar.

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

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

```
221 \newcommand*\Rlineflag{R}
222
```

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

```

223 \newcommand*{\linenumrepR}[1]{\@arabic{#1}}
224 \newcommand*{\sublinenumrepR}[1]{\@arabic{#1}}
225
\leftlinenumR \leftlinenumR and \rightlinenumR are the macros that are called to print the
\rightlinenumR right text's marginal line numbers. Much of the code for these is common and is
\l@dlinenumR maintained in \l@dlinenumR.
226 \newcommand*{\leftlinenumR}{%
227   \l@dlinenumR
228   \kern\linenumsep}
229 \newcommand*{\rightlinenumR}{%
230   \kern\linenumsep
231   \l@dlinenumR}
232 \newcommand*{\l@dlinenumR}{%
233   \numlabfont\linenumrepR{\line@numR}\Rlineflag%
234   \ifsublines@
235     \ifnum\subline@num>\z@
236       \unskip\fullstop\sublinenumrepR{\subline@numR}%
237       \fi
238   \fi}
239

```

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.

```

240 \newcount\line@numR
241 \newcount\subline@numR
242 \newcount\absline@numR
243

```

`\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.

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

```

244 \list@create{\line@listR}
245 \list@create{\insertlines@listR}
246 \list@create{\actionlines@listR}
247 \list@create{\actions@listR}
248

```

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

```

249 \list@create{\linesinpar@listL}
250 \list@create{\linesinpar@listR}
251 \list@create{\maxlinesinpar@list}
252
\page@numR The right text page number.
253 \newcount\page@numR
254
```

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.

```

255 \renewcommand*{\read@linelist}[1]{%
  We do do different things depending whether or not we are processing right text
256   \ifledRcol
257     \list@clear{\line@listR}%
258     \list@clear{\insertlines@listR}%
259     \list@clear{\actionlines@listR}%
260     \list@clear{\actions@listR}%
261     \list@clear{\linesinpar@listR}%
262     \list@clear{\linesonpage@listR}
263   \else
264     \list@clearing@reg
265     \list@clear{\linesinpar@listL}%
266     \list@clear{\linesonpage@listL}%
267   \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).
268   \list@clear{\maxlinesinpar@list}
  Now get the file and interpret it.
269   \get@linelistfile{#1}%
270   \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.

```

271 \ifledRcol
272   \global\page@numR=\m@ne
273   \ifx\actionlines@listR\empty
274     \gdef\next@actionlineR{1000000}%
275   \else
```

```

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

```

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.

```

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

```



```

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

```

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

```

\fix@page 341 \newcount\last@page@numR
342   \last@page@numR=-10000
343 \renewcommand*{\fix@page}[1]{%
344   \ifledRcol
345     \ifnum #1=\last@page@numR
346     \else
347       \ifbypage@R
348         \line@numR \z@ \subline@numR \z@
349       \fi
350       \page@numR=#1\relax
351       \last@page@numR=#1\relax
352       \def\next@page@numR{#1}%
353     \fi
354   \else
355     \ifnum #1=\last@page@num
356     \else
357       \ifbypage@

```

```

358     \line@num \z@ \subline@num \z@
359     \fi
360     \page@num=#1\relax
361     \last@page@num=#1\relax
362     \def\next@page@num{#1}%
363     \listcsxadd{normal@page@break}{\the\absline@num}
364     \fi
365     \fi}
366

```

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

```

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

```

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

```

399 \renewcommand*{\@set}[1]{%
400   \ifledRcol

```

```

401 \ifsublines@
402 \subline@numR=#1\relax
403 \else
404 \line@numR=#1\relax
405 \fi
406 \set@line@action
407 \else
408 \ifsublines@
409 \subline@num=#1\relax
410 \else
411 \line@num=#1\relax
412 \fi
413 \set@line@action
414 \fi}
415

```

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

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

```

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

```

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

```

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

```

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

```

436 \renewcommand*{\set@line@action}{%
437 \ifledRcol
438 \xright@appenditem{\the\absline@numR}\to\actionlines@listR
439 \ifsublines@

```

```

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

```

`\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.

```

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

```

`\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.

```

474 \newcount\@lockR
475 \newcount\sub@lockR
476
477 \newcommand*{\do@lockonR}{%
478   \xright@appenditem{\the\absline@numR}\to\actionlines@listR
479   \ifsublines@
480   \xright@appenditem{-1005}\to\actions@listR
481   \ifnum\sub@lockR=\z@

```

```

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

```

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

```

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

```

```

530 \ifledRcol
531   \do@lockoffR
532 \else
533   \do@lockoffL
534 \fi}
535 \global\let\lock@off=\do@lockoff
536

```

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

```

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

```

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

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

```

546   \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.

```

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

```

554   \begingroup

```

```

555 \let\@ref=\dummy@ref
556 \let\page@action=\relax
557 \let\sub@action=\relax
558 \let\set@line@action=\relax
559 \let\@lab=\relax
560 #2
561 \global\endpage@num=\page@numR
562 \global\endline@num=\line@numR
563 \global\endsubline@num=\subline@numR
564 \endgroup

```

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

```

565 \xright@appenditem%
566 {\the\page@numR|\the\line@numR|}%
567 \ifsublines@ \the\subline@numR \else 0\fi}%
568 \the\endpage@num|\the\endline@num|}%
569 \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.

```

570 #2
571 \else

```

And when not in right text

```

572 \@ref@reg{#1}{#2}%
573 \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.

```

574 \providecommand*\@pend}[1]{}
575 \renewcommand*\@pend}[1]{}%
576 \ifbypstart@\global\line@num=0\fi%
577 \xright@appenditem{#1}\to\linesinpar@listL}
578 \providecommand*\@pendR}[1]{}
579 \renewcommand*\@pendR}[1]{}%
580 \ifbypstart@R\global\line@numR=0\fi
581 \xright@appenditem{#1}\to\linesinpar@listR}
582

```

`\@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.

```

583 \providecommand*\@lopL}[1]{}
584 \renewcommand*\@lopL}[1]{}%
585 \xright@appenditem{#1}\to\linesonpage@listL}
586 \providecommand*\@lopR}[1]{}
587 \renewcommand*\@lopR}[1]{}%

```

```
588 \xright@appenditem{#1}\to\linesonpage@listR}
589
```

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`.
590 `\newwrite\linenum@outR`

`\iffirst@linenum@outR` Once any file is opened on this stream, we keep it open forever, or else switch to
`\first@linenum@outRtrue` another file that we keep open.
`\first@linenum@outRfalse` 591 `\newif\iffirst@linenum@outR`
592 `\first@linenum@outRtrue`

`\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.

```
593 \newcommand*{\line@list@stuffR}[1]{%
594   \read@linelist{#1}%
595   \iffirst@linenum@outR
596     \immediate\closeout\linenum@outR
597     \global\first@linenum@outRfalse
598     \immediate\openout\linenum@outR=#1
599   \else
600     \if@minipage%
601       \if@ledgroup%
602         \closeout\linenum@outR
603         \openout\linenum@outR=#1
604       \else
605         \immediate\closeout\linenum@outR
606         \immediate\openout\linenum@outR=#1\relax
607       \fi
608     \else
609       \closeout\linenum@outR%
610       \openout\linenum@outR=#1\relax%
611     \fi
612   \fi}
613
```

`\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.

```
614 \newcommand*{\new@lineL}{%
615   \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.

```

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

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

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

```

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

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

```

631 \renewcommand*{\advanceline}[1]{%
632   \ifledRcol \write\linenum@outR{\string\@adv[#1]}%
633   \else      \write\linenum@out{\string\@adv[#1]}%
634   \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.

```

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

```

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

```

648     \else          \write\linenum@out{\string\l@d@set[#1]} \fi
649   \fi}
650

```

`\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.

```

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

```

`\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.

```

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

```

15 Marking text for notes

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

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

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

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

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

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

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

```

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

```

\edtext And similarly for \edtext.

```

690 \renewcommand{\edtext}[2]{\leavevmode
691   \begingroup%
692     \renewcommand{\@tag}{\no@expands #1}%
693     \set@line%
694     \ifledRcol \global\insert@countR \z@%
695     \else      \global\insert@count \z@ \fi%
696     \ignorespaces #2\relax%
697     \@ifundefined{xpg@main@language}{%if not polyglossia
698       \flag@start}%
699       {\if@RTL\flag@end\else\flag@start\fi% be careful on the direction of writing with polyglossia
700       }%
701   \endgroup%
702   \showlemma{#1}%
703   \ifx\end@lemmas\empty \else%
704     \gl@p\end@lemmas\to\x@lemma%
705     \x@lemma%
706     \global\let\x@lemma=\relax%
707   \fi%
708   \@ifundefined{xpg@main@language}{%if not polyglossia
709     \flag@end}%
710     {\if@RTL\flag@start\else\flag@end\fi% be careful on the direction of writing with polyglossia
711     }%
712 }
713

```

`\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`.

```

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

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`

```

735 \newenvironment{pairs}{%
736   \l@dpairingtrue
737   \l@dpagingfalse
738   \initnumbering@sectcmd
739   \at@begin@pairs%
740 }{%
741   \l@dpairingfalse
742 }
743
```

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

```

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

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.

```

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

```

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

```

765 \newif\ifinstanzaL
766 \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.

```

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

```

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

```

\Rightsidehook 779 \newcommand*{\Leftsidehook}{}
\Rightsidehookend 780 \newcommand*{\Leftsidehookend}{}
781 \newcommand*{\Rightsidehook}{}
782 \newcommand*{\Rightsidehookend}{}

```

783

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.

```

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

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.

```

805 \newcount\num@linesR
806 \newbox\one@lineR
807 \newcount\par@lineR

```

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

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

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

```

808
809 \newcounter{pstartL}
810 \newcounter{pstartLold}
811 \renewcommand{\thepstartL}{\bfseries\@arabic\c@pstartL}. }
812 \newcounter{pstartR}
813 \newcounter{pstartRold}
814 \renewcommand{\thepstartR}{\bfseries\@arabic\c@pstartR}. }
815
816 \newcommand*{\pstartL}[1][1]{%
817   \if@nobreak%
818     \let\@oldnobreak\@nobreaktrue%
819   \else%
820     \let\@oldnobreak\@nobreakfalse%
821   \fi%
822   \@nobreaktrue%
823   \ifnumbering \else%
824     \led@err@PstartNotNumbered%
825     \beginnumbering%
826   \fi%
827   \ifnumberedpar%
828     \led@err@PstartInPstart%
829   \pend%
830 \fi%

```

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

```

831 \ifpstart@rtedL\else%
832   \setcounter{pstartLold}{\value{pstartL}}%
833   \list@clear{\inserts@list}%
834   \global\let\next@insert=\empty%
835   \global\pstart@rtedLtrue%
836 \fi%
837 \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.

```

838 \global\advance\l@dnumstartsL \@ne%
839 \ifnum\l@dnumstartsL>\l@dc@maxchunks%
840 \led@err@TooManyPstarts%
841 \global\l@dnumstartsL=\l@dc@maxchunks%
842 \fi%
843 \global\setnamebox{\l@dc@colrawbox\the\l@dnumstartsL}=\vbox\bgroup%
844 \ifautopar\else%
845 \ifnumberpstart%
846 \ifsidepstartnum%
847 \else%
848 \thepstartL%
849 \fi%
850 \fi%
851 \fi%
852 \hsize=\l@colwidth%
853 \numberedpar@true%
854 \iflabelpstart\protected@edef\@currentlabel%
855 {\p@pstartL\thepstartL}\fi%

Dump the optional arguments
856 \ifstrempy{#1}%
857 {}%
858 {\csgdef{before@pstartL@the\l@dnumstartsL}{\noindent#1}}%
859 }

860 \newcommandx*{\pstartR}[1][1]{%
861 \if@nbreak%
862 \let\@oldnbreak\@nbreaktrue%
863 \else%
864 \let\@oldnbreak\@nbreakfalse%
865 \fi%
866 \@nbreaktrue%
867 \ifnumberingR \else%
868 \led@err@PstartNotNumbered%
869 \beginnumberingR%
870 \fi%
871 \ifnumberedpar@%
872 \led@err@PstartInPstart%
873 \pendR%
874 \fi%
875 \ifpst@rtedR\else%
876 \setcounter{pstartRold}{\value{pstartR}}%
877 \list@clear{\inserts@listR}%
878 \global\let\next@insertR=\empty%
879 \global\pst@rtedRtrue%
880 \fi%
881 \begingroup\normal@pars%
882 \global\advance\l@dnumstartsR \@ne%

```



```

883 \ifnum\l@dnumpstartsR>\l@dc@maxchunks%
884   \led@err@TooManyPstarts%
885   \global\l@dnumpstartsR=\l@dc@maxchunks%
886 \fi%
887 \global\setnamebox{l@dRcolrawbox\the\l@dnumpstartsR}=\vbox\bgroup%
888   \ifautopar\else%
889     \ifnumberpstart%
890       \ifsidepstartnum\else%
891         \thepstartR%
892       \fi%
893     \fi%
894   \fi%
895 \hsize=\Rcolwidth%
896 \numberedpar@true%
897 \iflabelpstart\protected@edef\@currentlabel%
898   {\p@pstartR\thepstartR}\fi%
899 \ifstrempy{#1}%
900   {}
901   {\csgdef{before@pstartR@the\l@dnumpstartsR}{\noindent#1}}%
902 }

```

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

```

903 \newcommandx*\pendL}[1][1]{%
904   \ifnumbering \else%
905     \led@err@PendNotNumbered%
906   \fi%
907   \ifnumberedpar@ \else%
908     \led@err@PendNoPstart%
909   \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.

```

910 \l@dzeropenalties%
911 \endgraf\global\num@lines=\prevgraf\egroup%
912 \global\par@line=0%

```

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

```

913 \endgroup%
914 \ignorespaces%
915 \@oldnobreak%
916 \ifnumberpstart%
917   \addtocounter{pstartL}{1}%
918 \fi
919 \parledgroup@beforenotes@save{L}%

```

Dump content of the optional argument.

```

920 \ifstrempy{#1}%

```

```

921 {}%
922 {\csgdef{after@pendL@the\l@dnumpstartsL}{\noindent#1}}%
923 }

```

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

```

924 \newcommandx*{\pendR}[1][1]{%
925   \ifnumberingR \else%
926     \led@err@PendNotNumbered%
927   \fi%
928   \ifnumberedpar@ \else%
929     \led@err@PendNoPstart%
930   \fi%
931   \l@dzeropenalties%
932   \endgraf\global\num@linesR=\prevgraf\egroup%
933   \global\par@lineR=0%
934   \endgroup%
935   \ignorespaces%
936   \@oldnobreak%
937   \ifnumberpstart%
938     \addtocounter{pstartR}{1}%
939   \fi%
940   \parledgroup@beforenotes@save{R}%
941   \ifstrepty{#1}%
942     {}%
943     {\csgdef{after@pendR@the\l@dnumpstartsR}{\noindent#1}}%
944 }
945

```

17.2 Processing one line

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

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

```

946 \newbox\l@dleftbox
947 \newbox\l@drightbox
948

```

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

```

\countRline 949 \newcount\countLline
950   \countLline \z@
951 \newcount\countRline
952   \countRline \z@
953

```

`\@donereallinesL` We need to know the number of ‘real’ lines output (i.e., those that have been input by the user), and the total lines output (which includes any blank lines output for synchronisation).

`\@donereallinesR`

`\@donetotallinesR`

```

954 \newcount\@donereallinesL
955 \newcount\@donetotallinesL
956 \newcount\@donereallinesR
957 \newcount\@donetotallinesR
958

```

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

```

959 \newcommand*\do@lineL{%
960   \advance\countLline \@ne
961   \ifvbox\namebox{\l@dLcolrawbox\the\l@dpscL}%
962   {\vbadness=10000
963    \splittopskip=\z@
964    \do@lineLhook
965    \l@demptyd@ta
966    \global\setbox\one@line=\vsplit\namebox{\l@dLcolrawbox\the\l@dpscL}
967                                     to\baselineskip}%
968   \IfStrEq{\splitfirstmarks\parledgroup@}{begin}{\parledgroup@notes@startL}{%
969   \unvbox\one@line \global\setbox\one@line=\lastbox
970   \getline@numL
971   \ifnum\@lock>\@ne%
972     \inserthangingsymboltrue%
973   \else%
974     \inserthangingsymbolfalse%
975   \fi
976   \setbox\l@dleftbox
977   \hb@xt@ \Lcolwidth{%
978     \affixline@num
979     \xifinlist{\the\l@dpscL}{\eled@sections@@}%
980     {}%
981     {\print@lineL}}%
982   \add@penaltiesL
983   \global\advance\@donereallinesL\@ne
984   \global\advance\@donetotallinesL\@ne
985 \else
986   \setbox\l@dleftbox \hb@xt@ \Lcolwidth{\hspace*\Lcolwidth}%
987   \global\advance\@donetotallinesL\@ne
988 \fi}
989
990

```

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

```

991 \def\print@lineL{%
992   \affixpstart@numL%
993   \l@dld@ta %space kept for backward compatibility
994   \add@inserts\affixside@note%
995   \l@dlsn@te %space kept for backward compatibility
996   {\ledllfill\hb@xt@ \wd\one@line{\do@insidelinehook\inserthangingsymbolL\new@lineL\l@dunhbox@line
997     \l@drsn@te}}

```

998

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

```

999 \def\print@eledsectionL{%
1000   \add@inserts\affixside@note%
1001   \addtocounter{pstartL}{-1}%
1002   \ifdefstring{\@eledsectnotoc}{L}{\ledsectnotoc}{%
1003   \ifdefstring{\@eledsectmark}{L}{\ledsectnomark}%
1004   \numdef{\temp@}{\l@dpscl-1}%
1005   \xifinlist{\temp@}{\eled@sections@@}{\@nobreaktrue}{\@nobreakfalse}%
1006   \@eled@sectioningtrue%
1007   \csuse{eled@sectioning@the\l@dpscl}%
1008   \@eled@sectioningfalse%
1009   \global\csundef{eled@sectioning@the\l@dpscl}%
1010   \if@RTL%
1011     \hspace{-3\paperwidth}%
1012     {\hbox{\l@dunhbox@line{\one@line}} \new@line}%
1013   \else%
1014     \hspace{3\paperwidth}%
1015     {\new@line \hbox{\l@dunhbox@line{\one@line}}}%
1016   \fi%
1017   \vskip\eledsection@correcting@skip%
1018 }
1019
```

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

```

\doinlinedlineLhook 1020 \newcommand*{\dolineLhook}[1]{\gdef\do@lineLhook{#1}}%
\doinlinedlineRhook 1021 \newcommand*{\dolineRhook}[1]{\gdef\do@lineRhook{#1}}%
1022 \newcommand*{\doinlinedlineLhook}[1]{\gdef\do@insidelineLhook{#1}}%
1023 \newcommand*{\doinlinedlineRhook}[1]{\gdef\do@insidelineRhook{#1}}%
1024
```

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

```

\dolineLhook 1025 \newcommand*{\do@lineLhook}{%
\doinlinedlineLhook 1026 \newcommand*{\do@lineRhook}{%
\doinlinedlineRhook 1027 \newcommand*{\do@insidelineLhook}{%
1028 \newcommand*{\do@insidelineRhook}{%
1029
```

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

```

1030 \newcommand*{\do@lineR}{%
1031   \ledRcol@true%
1032   \advance\countRline \@ne
1033   \ifvbox\namebox{\l@dRcolrawbox\the\l@dpscl}%
1034   {\vbadness=10000
1035   \splittopskip=\z@
```

```

1036 \do@lineRhook
1037 \l@emptyd@ta
1038 \global\setbox\one@lineR=\vsplit\namebox{l@dRcolrawbox\the\l@dpscR}
1039 to\baselineskip}%
1040 \IfStrEq{\splitfirstmarks\parledgroup@}{begin}{\parledgroup@notes@startR}{\}
1041 \unvbox\one@lineR \global\setbox\one@lineR=\lastbox
1042 \getline@numR
1043 \ifnum\@lockR>\@ne%
1044 \inserthangingsymbolRtrue
1045 \else%
1046 \inserthangingsymbolRfalse%
1047 \fi%
1048 \setbox\l@dtrightbox
1049 \hb@xt@ \Rcolwidth{%
1050 \affixline@numR%
1051 \xifinlist{\the\l@dpscR}{\eled@sectionsR@@}%
1052 }%
1053 {\print@lineR}%
1054 }%
1055 \add@penaltiesR
1056 \global\advance\@donereallinesR\@ne
1057 \global\advance\@donetotallinesR\@ne
1058 \else
1059 \setbox\l@dtrightbox \hb@xt@ \Rcolwidth{\hspace*\Rcolwidth}}
1060 \global\advance\@donetotallinesR\@ne
1061 \fi
1062 \ledRcol@false%
1063 }
1064
1065

```

```

\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.

```

1066 \newcommand*{\getline@numR}{%
1067 \global\advance\absline@numR \@ne
1068 \do@actionsR
1069 \do@ballastR
1070 \ifledgroupnotesR\else\ifnumberline
1071 \ifsublines@
1072 \ifnum\sub@lockR<\tw@
1073 \global\advance\subline@numR \@ne
1074 \fi
1075 \else
1076 \ifnum\@lockR<\tw@
1077 \global\advance\line@numR \@ne

```

```

1078     \global\subline@numR \z@
1079     \fi
1080   \fi
1081 \fi
1082 \fi
1083 }
1084 \newcommand*{\getline@numL}{%
1085   \global\advance\absline@num \@ne
1086   \do@actions
1087   \do@ballast
1088   \ifledgroupnotesL@\else\ifnumberline
1089     \ifsublines@
1090       \ifnum\sub@lock<\tw@
1091         \global\advance\subline@num \@ne
1092       \fi
1093     \else
1094       \ifnum\@lock<\tw@
1095         \global\advance\line@num \@ne
1096         \global\subline@num \z@
1097       \fi
1098     \fi
1099 \fi
1100 \fi
1101 }
1102
1103

```

`\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.

```

1104 \newcommand*{\do@ballastR}{\global\ballast@count=\z@
1105   \begingroup
1106     \advance\absline@numR \@ne
1107     \ifnum\next@actionlineR=\absline@numR
1108       \ifnum\next@actionR>-1001
1109         \global\advance\ballast@count by -\c@ballast
1110       \fi
1111     \fi
1112   \endgroup}

```

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

`\do@actions@fixedcodeR`

`\do@actions@nextR`

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

```

1113 \newcommand*{\do@actions@fixedcodeR}{%
1114   \ifcase\@l@temptcnta%
1115     \or% % 1001
1116     \global\sublines@true
1117     \or% % 1002

```

```

1118 \global\sublines@false
1119 \or% % 1003
1120 \global\@lockR=\@ne
1121 \or% % 1004
1122 \ifnum\@lockR=\tw@
1123 \global\@lockR=\thr@@
1124 \else
1125 \global\@lockR=\z@
1126 \fi
1127 \or% % 1005
1128 \global\sub@lockR=\@ne
1129 \or% % 1006
1130 \ifnum\sub@lockR=\tw@
1131 \global\sub@lockR=\thr@@
1132 \else
1133 \global\sub@lockR=\z@
1134 \fi
1135 \or% % 1007
1136 \l@dskipnumbertrue
1137 \else
1138 \led@warn@BadAction
1139 \fi}
1140
1141
1142 \newcommand*{\do@actionsR}{%
1143 \global\let\do@actions@nextR=\relax
1144 \@l@dttempcntb=\absline@numR
1145 \ifnum\@l@dttempcntb<\next@actionlineR\else
1146 \ifnum\next@actionR>-1001\relax
1147 \global\page@numR=\next@actionR
1148 \ifbypage@R
1149 \global\line@numR \z@ \global\subline@numR \z@
1150 \fi
1151 \else
1152 \ifnum\next@actionR<-4999\relax % 9/05 added relax here
1153 \@l@dttempcnta=-\next@actionR
1154 \advance\@l@dttempcnta by -5001\relax
1155 \ifsublines@
1156 \global\subline@numR=\@l@dttempcnta
1157 \else
1158 \global\line@numR=\@l@dttempcnta
1159 \fi
1160 \else
1161 \@l@dttempcnta=-\next@actionR
1162 \advance\@l@dttempcnta by -1000\relax
1163 \do@actions@fixedcodeR
1164 \fi
1165 \fi
1166 \ifx\actionlines@listR\empty
1167 \gdef\next@actionlineR{1000000}%

```

```

1168 \else
1169 \glp\actionlines@listR\to\next@actionlineR
1170 \glp\actions@listR\to\next@actionR
1171 \global\let\do@actions@nextR=\do@actionsR
1172 \fi
1173 \fi
1174 \do@actions@nextR}
1175

```

17.4 Line number printing

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

```

\ch@cksub@l@ckR 1176
\ch@ck@l@ckR 1177 \providecommand*{\l@dcalcnum}[3]{%
\fx@l@cksR 1178 \ifnum #1 > #2\relax
\affixline@numR 1179 \l@dtempcnta = #1\relax
1180 \advance\l@dtempcnta by -#2\relax
1181 \divide\l@dtempcnta by #3\relax
1182 \multiply\l@dtempcnta by #3\relax
1183 \advance\l@dtempcnta by #2\relax
1184 \else
1185 \l@dtempcnta=#2\relax
1186 \fi}
1187
1188 \newcommand*{\ch@cksub@l@ckR}{%
1189 \ifcase\sub@lockR
1190 \or
1191 \ifnum\sublock@disp=\@ne
1192 \l@dtempcntb \z@ \l@dtempcnta \@ne
1193 \fi
1194 \or
1195 \ifnum\sublock@disp=\tw@
1196 \else
1197 \l@dtempcntb \z@ \l@dtempcnta \@ne
1198 \fi
1199 \or
1200 \ifnum\sublock@disp=\z@
1201 \l@dtempcntb \z@ \l@dtempcnta \@ne
1202 \fi
1203 \fi}
1204
1205 \newcommand*{\ch@ck@l@ckR}{%
1206 \ifcase\l@lockR
1207 \or
1208 \ifnum\lock@disp=\@ne
1209 \l@dtempcntb \z@ \l@dtempcnta \@ne
1210 \fi
1211 \or
1212 \ifnum\lock@disp=\tw@

```



```

1213 \else
1214 \l@dttempcntb \z@ \l@dttempcnta \@ne
1215 \fi
1216 \or
1217 \ifnum\lock@disp=\z@
1218 \l@dttempcntb \z@ \l@dttempcnta \@ne
1219 \fi
1220 \fi}
1221
1222 \newcommand*{\f@x@l@cksR}{%
1223 \ifcase\@lockR
1224 \or
1225 \global\@lockR \tw@
1226 \or \or
1227 \global\@lockR \z@
1228 \fi
1229 \ifcase\sub@lockR
1230 \or
1231 \global\sub@lockR \tw@
1232 \or \or
1233 \global\sub@lockR \z@
1234 \fi}
1235
1236
1237 \newcommand*{\affixline@numR}{%
1238 \ifl@edgroupnotesR\else\ifnumberline
1239 \ifl@dskipnumber
1240 \global\l@dskipnumberfalse
1241 \else
1242 \ifsublines@
1243 \l@dttempcntb=\subline@numR
1244 \l@dcalcnum{\subline@numR}{\c@firstsublinenumR}{\c@sublinenumincrementR}%
1245 \ch@cksub@lockR
1246 \else
1247 \l@dttempcntb=\line@numR
1248 \ifx\linenumberlist\empty
1249 \l@dcalcnum{\line@numR}{\c@firstlinenumR}{\c@linenumincrementR}%
1250 \else
1251 \l@dttempcnta=\line@numR
1252 \edef\rem@inder{\linenumberlist,\number\line@numR,}%
1253 \edef\sc@n@list{\def\noexpand\sc@n@list
1254 ###1,\number\l@dttempcnta,###2|{\def\noexpand\rem@inder{###2}}}%
1255 \sc@n@list\expandafter\sc@n@list\rem@inder|
1256 \ifx\rem@inder\empty\advance\l@dttempcnta\@ne\fi
1257 \fi
1258 \ch@ck@l@ckR
1259 \fi
1260 \ifnum\l@dttempcnta=\l@dttempcntb
1261 \if@twocolumn
1262 \if@firstcolumn

```

```

1263     \gdef\l@dld@ta{\llap{\leftlinenumR}}}%
1264     \else
1265     \gdef\l@drd@ta{\rlap{\rightlinenumR}}}%
1266     \fi
1267   \else
1268     \@l@tempcntb=\line@marginR
1269     \ifnum\@l@tempcntb>\@ne
1270       \advance\@l@tempcntb by\page@numR
1271     \fi
1272     \ifodd\@l@tempcntb
1273       \gdef\l@drd@ta{\rlap{\rightlinenumR}}}%
1274     \else
1275       \gdef\l@dld@ta{\llap{\leftlinenumR}}}%
1276     \fi
1277   \fi
1278 \fi
1279 \f@x@l@cksR
1280 \fi
1281 \fi
1282 \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 1283
  \leftpstartnumR 1284 \newcommand*{\affixpstart@numL}{%
\rightpstartnumR 1285 \ifsidepstartnum
\leftpstartnumL 1286 \if@twocolumn
\rightpstartnumL 1287   \if@firstcolumn
  \ifpstartnumR 1288     \gdef\l@dld@ta{\llap{\leftpstartnumL}}}%
1289     \else
1290     \gdef\l@drd@ta{\rlap{\rightpstartnumL}}}%
1291     \fi
1292   \else
1293     \@l@tempcntb=\line@margin
1294     \ifnum\@l@tempcntb>\@ne
1295       \advance\@l@tempcntb \page@num
1296     \fi
1297     \ifodd\@l@tempcntb
1298       \gdef\l@drd@ta{\rlap{\rightpstartnumL}}}%
1299     \else
1300       \gdef\l@dld@ta{\llap{\leftpstartnumL}}}%

```

```

1301     \fi
1302     \fi
1303 \fi
1304 }
1305 \newcommand*{\affixpstart@numR}{%
1306 \ifsidepstartnum
1307 \if@twocolumn
1308     \if@firstcolumn
1309         \gdef\l@dld@ta{\llap{\leftpstartnumR}}}%
1310     \else
1311         \gdef\l@drd@ta{\rlap{\rightpstartnumR}}}%
1312     \fi
1313 \else
1314     \l@dtempcntb=\line@marginR
1315     \ifnum\l@dtempcntb>\@ne
1316         \advance\l@dtempcntb \page@numR
1317     \fi
1318     \ifodd\l@dtempcntb
1319         \gdef\l@drd@ta{\rlap{\rightpstartnumR}}}%
1320     \else
1321         \gdef\l@dld@ta{\llap{\leftpstartnumR}}}%
1322     \fi
1323 \fi
1324 \fi
1325 }
1326
1327 \newcommand*{\leftpstartnumL}{
1328 \ifpstartnum
1329 \thepstartL
1330 \kern\linenumsep\global\pstartnumfalse\fi
1331 }
1332 \newcommand*{\rightpstartnumL}{
1333 \ifpstartnum\kern\linenumsep
1334 \thepstartL
1335 \global\pstartnumfalse\fi
1336 }
1337 \newif\ifpstartnumR
1338 \pstartnumRtrue
1339 \newcommand*{\leftpstartnumR}{
1340 \ifpstartnumR
1341 \thepstartR
1342 \kern\linenumsep\global\pstartnumRfalse\fi
1343 }
1344 \newcommand*{\rightpstartnumR}{
1345 \ifpstartnumR\kern\linenumsep
1346 \thepstartR
1347 \global\pstartnumRfalse\fi
1348 }

```

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.

```
1349 \list@create{\inserts@listR}
```

`\add@insertsR` The right text version.

```
\add@inserts@nextR 1350 \newcommand*{\add@insertsR}{%
1351   \global\let\add@inserts@nextR=\relax
1352   \ifx\inserts@listR\empty \else
1353     \ifx\next@insertR\empty
1354       \ifx\insertlines@listR\empty
1355         \global\noteschanged@true
1356         \gdef\next@insertR{100000}%
1357       \else
1358         \gl@p\insertlines@listR\to\next@insertR
1359       \fi
1360     \fi
1361     \ifnum\next@insertR=\absline@numR
1362       \gl@p\inserts@listR\to\@insertR
1363       \@insertR
1364       \global\let\@insertR=\undefined
1365       \global\let\next@insertR=\empty
1366       \global\let\add@inserts@nextR=\add@insertsR
1367     \fi
1368   \fi
1369   \add@inserts@nextR}
1370
```

17.7 Penalties

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

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

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

```

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

```

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

```

1371 \newcommand*{\add@penaltiesL}{ }
1372 \newcommand*{\add@penaltiesR}{ }
1373

```

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.

```

1374 \newcommand*{\flush@notesR}{%
1375   \xloop
1376   \ifx\inserts@listR\empty \else
1377     \gl@p\inserts@listR\to\@insertR
1378     \@insertR
1379     \global\let\@insertR=\undefined
1380   \repeat}
1381

```

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
1382 \def\printlinesR#1|#2|#3|#4|#5|#6|#7|{\begingroup
1383 \setprintlines{#1}{#2}{#3}{#4}{#5}{#6}%
1384 \ifl@d@pnum #1\fullstop\fi
1385 \ifledplinenum \linenumr@p{#2}\Rlineflag\else \symlinenum\fi
1386 \ifl@d@ssub \fullstop \sublinenumr@p{#3}\fi
1387 \ifl@d@dash \endashchar\fi
1388 \ifl@d@pnum #4\fullstop\fi
1389 \ifl@d@elin \linenumr@p{#5}\Rlineflag\fi
1390 \ifl@d@esl \ifl@d@elin \fullstop\fi \sublinenumr@p{#6}\fi
1391 \endgroup}
1392
1393 \let\ledsavedprintlines\printlines
1394

```

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.

```

1395 \list@create{\labelref@listR}
1396

```

`\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.

```

1397 \renewcommand*{\edlabel}[1]{\@bsphack
1398 \ifledRcol
1399 \write\linenum@outR{\string\@lab}%
1400 \ifx\labelref@listR\empty
1401 \xdef\label@refs{\zz@@@}%
1402 \else
1403 \gl@p\labelref@listR\to\label@refs
1404 \fi
1405 \ifvmode
1406 \advancelabel@refs
1407 \fi
1408 \protected@write\@auxout{%
1409 {\string\l@dmake@labelsR\space\thepage|\label@refs|\the\c@pstartR|{#1}}%

```

```

1410 \else
1411   \write\linenum@out{\string\@lab}%
1412   \ifx\labelref@list\empty
1413     \xdef\label@refs{\zz@@@}%
1414   \else
1415     \gl@p\labelref@list\to\label@refs
1416   \fi
1417   \ifvmode
1418     \advancelabel@refs
1419   \fi
1420 \protected@write\@auxout{%
1421   {\string\l@dmake@labels\space\thepage|\label@refs|\the\c@pstart|{#1}}%
1422 \fi
1423 \@esphack}
1424
1425

```

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

```

1426 \def\l@dmake@labelsR#1|#2|#3|#4|#5{%
1427   \expandafter\ifx\csname the@label#5\endcsname \relax\else
1428     \led@warn@DuplicateLabel{#4}%
1429   \fi
1430   \expandafter\gdef\csname the@label#5\endcsname{#1|#2\Rlineflag|#3|#4}%
1431   \ignorespaces}
1432 \AtBeginDocument{%
1433   \def\l@dmake@labelsR#1|#2|#3|#4|#5{%
1434   }
1435

```

`\@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.

```

1436 \renewcommand*{\@lab}{%
1437   \ifledRcol
1438     \xright@appenditem{\linenumr@p{\line@numR}}{%
1439       \ifsublines@ \sublinenumr@p{\subline@numR}\else 0\fi}%
1440     \to\labelref@listR
1441   \else
1442     \xright@appenditem{\linenumr@p{\line@num}}{%
1443       \ifsublines@ \sublinenumr@p{\subline@num}\else 0\fi}%
1444     \to\labelref@list
1445   \fi}
1446

```

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* 1447 \WithSuffix\newcommand\sidenotemargin*[1]{%
1448   \l@dgetsidenote@margin{#1}
1449   \global\sidenote@marginR=\@l@dttempcntb
1450   \global\sidenote@margin=\@l@dttempcntb
1451 }
1452 \newcount\sidenote@marginR
1453 \global\sidenote@margin=\@ne
1454
```

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

```
1455 \newcommand*\affixside@noteR{%
1456   \def\sidenotecontent@{%
1457     \numgdef{\itemcount@}{0}%
1458     \def\do##1{%
1459       \ifnumequal{\itemcount@}{0}%
1460         {%
1461           \appto\sidenotecontent@{##1}}% Not print not separator before the 1st note
1462           {\appto\sidenotecontent@{\sidenotesep ##1}%
1463           }%
1464           \numgdef{\itemcount@}{\itemcount@+1}%
1465         }%
1466       \dolistloop{\l@dcsnotetext}%
1467       \ifnumgreater{\itemcount@}{1}{\led@err@ManySidenotes}{}%
1468     \gdef\@templ@d{%
1469     \gdef\@templ@n{\l@dcsnotetext\l@dcsnotetext@l\l@dcsnotetext@r}%
1470     \ifx\@templ@d\@templ@n \else%
1471       \if@twocolumn%
1472         \if@firstcolumn%
1473           \setl@dlp@rbox{##1}{\sidenotecontent@}%
1474         \else%
1475           \setl@drp@rbox{\sidenotecontent@}%
1476         \fi%
1477       \else%
1478         \@l@dttempcntb=\sidenote@marginR%
1479         \ifnum\@l@dttempcntb>\@ne%
1480           \advance\@l@dttempcntb by\page@numR%
1481         \fi%
1482         \ifodd\@l@dttempcntb%
1483           \setl@drp@rbox{\sidenotecontent@}%
1484         \gdef\sidenotecontent@{%
1485         \numdef{\itemcount@}{0}%
1486         \dolistloop{\l@dcsnotetext@l}%
1487         \ifnumgreater{\itemcount@}{1}{\led@err@ManyLeftnotes}{}%

```



```

1488     \setl@dlp@rbox{\sidenotecontent@}%
1489     \else%
1490     \setl@dlp@rbox{\sidenotecontent@}%
1491     \gdef\sidenotecontent@{}%
1492     \numdef\itemcount@{0}%
1493     \dolistloop{\l@dcstotetext@r}%
1494     \ifnumgreater{\itemcount@}{1}{\led@err@ManyRightnotes}{}%
1495     \setl@drp@rbox{\sidenotecontent@}%
1496     \fi%
1497 \fi%
1498 \fi%
1499 }
1500

```

21 Familiar footnotes

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

```

1501 \renewcommand{\l@dbfnote}[1]{%
1502   \ifnumberedpar@
1503   \gdef\@tag{#1}%
1504   \ifledRcol%
1505     \xright@appenditem{\noexpand\v1@dbfnote{\csexpandonce{\@tag}}{\@thefnmark}}%
1506                     \to\inserts@listR
1507     \global\advance\insert@countR \@ne%
1508   \else%
1509     \xright@appenditem{\noexpand\v1@dbfnote{\csexpandonce{\@tag}}{\@thefnmark}}%
1510                     \to\inserts@list
1511     \global\advance\insert@count \@ne%
1512   \fi
1513 \fi\ignorespaces}
1514

```

`\normalbfnoteX`

```

1515 \renewcommand{\normalbfnoteX}[2]{%
1516   \ifnumberedpar@
1517   \ifledRcol%
1518     \ifluatex
1519       \footnotelang@lua[R]%
1520     \fi
1521     \ifundefined{xpg@main@language}%if polyglossia
1522     {%
1523       \footnotelang@poly[R]}%
1524     \protected@csxdef{thisfootnote}{\csuse{thefootnote#1}}%
1525     \xright@appenditem{\noexpand\vbfnoteX{#1}{#2}{\csexpandonce{thisfootnote}}}%
1526                     \to\inserts@listR
1527     \global\advance\insert@countR \@ne%
1528   \else%

```

```

1529 \ifluatex
1530 \footnotelang@lua%
1531 \fi
1532 \@ifundefined{xpg@main@language}%if polyglossia
1533 {}%
1534 {\footnotelang@poly}%
1535 \protected@csxdef{thisfootnote}{\csuse{thefootnote#1}}%
1536 \xright@appenditem{\noexpand\vbfnoteX{#1}{#2}{\csexpandonce{thisfootnote}}}%
1537 \to\inserts@list
1538 \global\advance\insert@count \@ne%
1539 \fi
1540 \fi\ignorespaces}
1541

```

22 Verse

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

```

\inserthangingsymbolL
\inserthangingsymbolR 1542 \newif\ifinserthangingsymbolR
1543 \newcommand{\inserthangingsymbolL}{%
1544 \ifinserthangingsymbol%
1545 \ifinstanzaL%
1546 \hangingsymbol%
1547 \fi%
1548 \fi}
1549 \newcommand{\inserthangingsymbolR}{%
1550 \ifinserthangingsymbolR%
1551 \ifinstanzaR%
1552 \hangingsymbol%
1553 \fi%
1554 \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 1555 \newcommand{\correcthangingL}{%
1556 \ifl@dpaging\else%
1557 \ifinstanzaL%
1558 \ifinserthangingsymbol%
1559 \hskip \@ifundefined{sza@00}{0}{\expandafter%
1560 \noexpand\csname sza@00\endcsname}\stanzaindentbase%
1561 \fi%
1562 \fi%
1563 \fi}

```

```

1564
1565 \newcommand{\correcthangingR}{%
1566 \ifl@dpaging\else%
1567   \ifinstanzaR%
1568     \ifinserthangingsymbolR%
1569       \hskip \@ifundefined{sza@00}{0}{\expandafter%
1570         \noexpand\csname sza@00\endcsname}\stanzaindentbase%
1571     \fi%
1572   \fi%
1573 \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.

```

1574 \chardef\next=\catcode'\&
1575 \catcode'\&=\active
1576

```

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

```

1577 \newenvironment{astanza}{%
1578 \startstanzahook
1579 \catcode'\&\active
1580 \global\stanza@count\@ne\stanza@modulo\@ne
1581 \ifnum\usernamecount{sza@00}=\z@
1582   \let\stanza@hang\relax
1583   \let\endlock\relax
1584 \else
1585 %%% \interlinepenalty\@M % this screws things up, but I don't know why
1586 \rightskip\z@ plus 1fil\relax
1587 \fi
1588 \ifnum\usernamecount{szp@00}=\z@
1589   \let\sza@penalty\relax
1590 \fi
1591 \def&{%
1592   \endlock\mbox{}}%
1593 \sza@penalty
1594 \global\advance\stanza@count\@ne
1595 \@astanza@line}%
1596 \def\&{%
1597   \endlock\mbox{}}
1598 \pend
1599 \endstanzaextra}%
1600 \pstart
1601 \@astanza@line
1602 }{}
1603

```

`\@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.

```

1604 \newcommand*{\@astanza@line}{%
1605   \ifnum\value{stanzaindentsrepetition}=0
1606     \parindent=\csname sza@\number\stanza@count
1607       @\endcsname\stanzaindentbase
1608   \else
1609     \parindent=\csname sza@\number\stanza@modulo
1610       @\endcsname\stanzaindentbase
1611     \managestanza@modulo
1612   \fi
1613   \par
1614   \stanza@hang%\mbox{}%
1615   \ignorespaces}
1616

```

Lastly reset the modified category codes.

```

1617 \catcode'\&=\next
1618

```

23 Naming macros

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

```

\newnamebox A set of macros for creating and using ‘named’ boxes; the macros are called after
\setnamebox the regular box macros, but including the string ‘name’.
\unhnamebox 1619 \providecommand*{\newnamebox}[1]{%
\unvnamebox 1620   \expandafter\newbox\csname #1\endcsname}
\namebox 1621 \providecommand*{\setnamebox}[1]{%
1622   \expandafter\setbox\csname #1\endcsname}
1623 \providecommand*{\unhnamebox}[1]{%
1624   \expandafter\unhbox\csname #1\endcsname}
1625 \providecommand*{\unvnamebox}[1]{%
1626   \expandafter\unvbox\csname #1\endcsname}
1627 \providecommand*{\namebox}[1]{%
1628   \csname #1\endcsname}
1629

\newnamecount Macros for creating and using ‘named’ counts.
\usenamecount 1630 \providecommand*{\newnamecount}[1]{%
1631   \expandafter\newcount\csname #1\endcsname}
1632 \providecommand*{\usenamecount}[1]{%
1633   \csname #1\endcsname}
1634

```

24 Counts and boxes for parallel texts

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

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

```
1635 \newcount\l@dc@maxchunks
1636 \newcommand{\maxchunks}[1]{\l@dc@maxchunks=#1}
1637   \maxchunks{5120}
1638
```

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

`\l@dnumpstartsR` 1639 \newcount\l@dnumpstartsR
 1640

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

`\l@pscR` 1641 \newcount\l@dpscL
 1642 \newcount\l@dpscR
 1643

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

```
1644 \newcommand*{\l@dsetuprawboxes}{%
1645   \@l@dttempcntb=\l@dc@maxchunks
1646   \loop\ifnum\@l@dttempcntb>\z@
1647     \newnamebox{\l@dLcolrawbox\the\@l@dttempcntb}
1648     \newnamebox{\l@dRcolrawbox\the\@l@dttempcntb}
1649     \advance\@l@dttempcntb \m@ne
1650   \repeat}
1651
```

`\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.

```
1652 \newcommand*{\l@dsetupmaxlinecounts}{%
1653   \@l@dttempcntb=\l@dc@maxchunks
1654   \loop\ifnum\@l@dttempcntb>\z@
1655     \newnamecount{\l@dmaxlinesinpar\the\@l@dttempcntb}
1656     \advance\@l@dttempcntb \m@ne
1657   \repeat}
1658 \newcommand*{\l@dzeromaxlinecounts}{%
1659   \begingroup
1660   \@l@dttempcntb=\l@dc@maxchunks
1661   \loop\ifnum\@l@dttempcntb>\z@
```

```

1662 \global\usernamecount{l@dmxlinesinpar\the\@l@dttempcntb}=\z@
1663 \advance\@l@dttempcntb \m@ne
1664 \repeat
1665 \endgroup}
1666

```

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

```

1667 \AtBeginDocument{%
1668 \l@dsetuprawboxes
1669 \l@dsetupmaxlinecounts
1670 \l@dzeromaxlinecounts
1671 \l@dnumpstartsL=\z@
1672 \l@dnumpstartsR=\z@
1673 \l@dpscL=\z@
1674 \l@dpscR=\z@}
1675

```

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 1676 \newif\ifl@dusedbabel
\l@dusedbabeltrue 1677 \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 \.

```

```

1678 \newcommand*{\l@dbbl@set@language}[1]{%
1679 \edef\language{#1}%
1680 \select@language{\language}%

```

```

1681 \if@filesw
1682   \protected@write\@auxout{}\string\select@language{\language}%
1683   \addtocontents{toc}{\string\select@language{\language}}%
1684   \addtocontents{lof}{\string\select@language{\language}}%
1685   \addtocontents{lot}{\string\select@language{\language}}%
1686 \fi}
1687

```

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 1688 \providecommand{\selectlanguage}[1]{%
1689 \newcommand*{\l@duselanguage}[1]{%
1690 \gdef\theledlanguageL{
1691 \gdef\theledlanguageR{
1692

```

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

```

1693 \AtBeginDocument{%
1694   \ifundefined{xpg@main@language}{%
1695     \ifundefined{bbl@main@language}{%

```

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

```

1696   \l@dusedbabelfalse
1697   \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.

```

1698   \l@dusedbabeltrue
1699   \let\l@doldselectlanguage\selectlanguage
1700   \let\l@doldbbl@set@language\bbl@set@language
1701   \let\bbl@set@language\l@dbbl@set@language
1702   \renewcommand{\selectlanguage}[1]{%
1703     \l@doldselectlanguage{#1}%
1704     \ifledRcol \gdef\theledlanguageR{#1}%
1705     \else      \gdef\theledlanguageL{#1}%
1706   \fi}

```

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

```

1707   \renewcommand*{\l@duselanguage}[1]{%
1708     \l@doldselectlanguage{#1}}

```

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

```

1709   \gdef\theledlanguageL{\bbl@main@language}%
1710   \gdef\theledlanguageR{\bbl@main@language}%

```

```

1711 }%
1712 }
    If on Polyglossia
1713 { \let\old@otherlanguage\otherlanguage%
1714   \renewcommand{\otherlanguage}[2][]{%
1715     \selectlanguage[#1]{#2}%
1716     \ifledRcol \gdef\theledlanguageR{#2}%
1717     \else      \gdef\theledlanguageL{#2}%
1718     \fi}%
1719   \let\l@duselanguage\select@language%
1720   \gdef\theledlanguageL{\xpg@main@language}%
1721   \gdef\theledlanguageR{\xpg@main@language}%
    That's it.
1722 }}

```

```

    \if@pstarts \check@pstarts returns \@pstartstrue if there are any unprocessed chunks.
    \@pstartstrue 1723 \newif\if@pstarts
    \@pstartsfalse 1724 \newcommand*\check@pstarts}{%
    \check@pstarts 1725 \@pstartsfalse
    1726 \ifnum\l@dnumpstartsL>\l@dpscL
    1727 \@pstartstrue
    1728 \else
    1729 \ifnum\l@dnumpstartsR>\l@dpscR
    1730 \@pstartstrue
    1731 \fi
    1732 \fi
    1733 }
    1734

```

```

    \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 1735 \newif\ifaraw@text
    1736 \araw@textfalse
    1737 \newcommand*\checkraw@text}{%
    1738 \araw@textfalse
    1739 \ifvbox\namebox{\l@dLcolrawbox\the\l@dpscL}
    1740 \araw@texttrue
    1741 \else
    1742 \ifvbox\namebox{\l@dRcolrawbox\the\l@dpscR}
    1743 \araw@texttrue
    1744 \fi
    1745 \fi
    1746 }
    1747

```

```

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

```



```

1748 \newcommand*{\@writelinesinparL}{%
1749   \edef\next{%
1750     \write\linenum@out{\string\@pend[\the\@donereallinesL]}}%
1751   \next
1752   \global\@donereallinesL \z@}
1753 \newcommand*{\@writelinesinparR}{%
1754   \edef\next{%
1755     \write\linenum@outR{\string\@pendR[\the\@donereallinesR]}}%
1756   \next
1757   \global\@donereallinesR \z@}
1758

```

26 Parallel columns

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

```

\@eledsectionR 1759 \newsavebox{\@eledsectionL}%
1760 \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.

```

1761 \newcommand*{\Columns}{%
1762   \eledsection@correcting@skip=-\baselineskip% Correction for sections' titles
1763   \setcounter{pstartL}{\value{pstartLold}}
1764   \setcounter{pstartR}{\value{pstartRold}}
1765   \ifnum\l@dnumpstartsL=\l@dnumpstartsR\else
1766     \led@err@BadLeftRightPstarts{\the\l@dnumpstartsL}{\the\l@dnumpstartsR}%
1767   \fi

```

Start a group and zero counters, etc.

```

1768 \begingroup
1769   \l@dzeropenalties
1770   \endgraf\global\num@lines=\prevgraf
1771   \global\num@linesR=\prevgraf
1772   \global\par@line=\z@
1773   \global\par@lineR=\z@
1774   \global\l@dpscL=\z@
1775   \global\l@dpscR=\z@

```

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

```

1776   \check@pstarts
1777   \loop\if@pstarts
1778     \global\pstartnumtrue
1779     \global\pstartnumRtrue

```

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

```

1780   \global\advance\l@dpscL \@ne
1781   \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

```

1782      \checkraw@text
1783 {      \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.

1784      \l@duselanguage{\theledlanguageL}%
1785      \do@lineL
1786      \xifinlist{\the\l@dpscL}{\eled@sections@}
1787      {%
1788      \ifdefstring{\@eledsectmark}{L}%
1789      {\csuse{eled@sectmark@the\l@dpscL}%
1790      }{}}%
1791      \global\csundef{eled@sectmark@the\l@dpscL}%
1792      \savebox{\@eledsectionL}{\parbox[t]{}[t]{\Lcolwidth}{\vbox{}}\print@eledse
1793      }%
1794      {}%
1795      \l@duselanguage{\theledlanguageR}%
1796      \do@lineR
1797      \xifinlist{\the\l@dpscR}{\eled@sectionsR@}
1798      {%
1799      \ifdefstring{\@eledsectmark}{R}%
1800      {\csuse{eled@sectmark@the\l@dpscR }%
1801      }{}}%
1802      \global\csundef{eled@sectmark@the\l@dpscR }%
1803      \savebox{\@eledsectionR}{\parbox[t]{}[t]{\Rcolwidth}{\vbox{}}\print@eledse
1804      }%
1805      \hb@xt@ \hsize{%
1806      \ifdefstring{\columns@position}{L}{-}{\hfill }%
1807      \unhbox\l@dleftbox%
1808      \ifhbox\@eledsectionL%
1809      \usebox{\@eledsectionL}%
1810      \fi%
1811      \print@columnseparator%
1812      \unhbox\l@drightbox%
1813      \ifhbox\@eledsectionR%
1814      \usebox{\@eledsectionR}%
1815      \fi%
1816      \ifdefstring{\columns@position}{R}{-}{\hfill}%
1817      }%
1818      \checkraw@text
1819      \checkverseL
1820      \checkverseR
1821      \checkpb@columns
1822      \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.

```

1823 \@writelinesinparL
1824 \@writelinesinparR
1825 \check@pstarts
1826 \ifbypstart@
1827 \write\linenum@out{\string\@set[1]}
1828 \resetprevline@
1829 \fi
1830 \ifbypstart@R
1831 \write\linenum@outR{\string\@set[1]}
1832 \resetprevline@
1833 \fi
1834 \addtocounter{pstartL}{1}
1835 \addtocounter{pstartR}{1}
1836 \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.

```

1837 \flush@notes
1838 \flush@notesR
1839 \endgroup
1840 \global\l@dpscL=\z@
1841 \global\l@dpscR=\z@
1842 \global\l@dnumpstartsL=\z@
1843 \global\l@dnumpstartsR=\z@
1844 \ignorespaces
1845 \global\instanzaLfalse
1846 \global\instanzaRfalse}
1847

```

`\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).

```

1848 \def\print@columnseparator{%
1849 \ifdim\beforecolumnseparator<0pt%
1850 \hfill%
1851 \else%
1852 \hspace{\beforecolumnseparator}%
1853 \fi%
1854 \columnseparator%
1855 \ifdim\aftercolumnseparator<0pt%
1856 \hfill%
1857 \else%
1858 \hspace{\beforecolumnseparator}%
1859 \fi%
1860 }%
1861 %\end{macrocode}
1862 % \end{macro}
1863 % \begin{macro}{\checkpb@columns}
1864 % \cs{checkpb@columns} prevent or make pagebreaking in columns, depending of the use of \cs{ledpb} on

```

```

1865 %      \begin{macrocode}
1866
1867 \newcommand{\checkpb@columns}{%
1868     \newif\if@pb
1869     \newif\if@nopb
1870     \IfStrEq{\led@pb@setting}{before}{
1871         \numdef{\next@absline}{\the\absline@num+1}%
1872         \numdef{\next@abslineR}{\the\absline@numR+1}%
1873         \xifinlistcs{\next@absline}{l@prev@pb}{\@pbtrue}{}%
1874         \xifinlistcs{\next@abslineR}{l@prev@pbR}{\@pbtrue}{%
1875         \xifinlistcs{\next@absline}{l@prev@nopb}{\@nopbtrue}{}%
1876         \xifinlistcs{\next@abslineR}{l@prev@nopbR}{\@nopbtrue}{%
1877     }{}
1878     \IfStrEq{\led@pb@setting}{after}{
1879         \xifinlistcs{\the\absline@num}{l@prev@pb}{\@pbtrue}{}%
1880         \xifinlistcs{\the\absline@numR}{l@prev@pbR}{\@pbtrue}{%
1881         \xifinlistcs{\the\absline@num}{l@prev@nopb}{\@nopbtrue}{}%
1882         \xifinlistcs{\the\absline@numR}{l@prev@nopbR}{\@nopbtrue}{%
1883     }{}
1884 \if@nopb\nopagebreak[4]\enlargethispage{\baselineskip}\fi
1885 \if@pb\pagebreak[4]\fi
1886 }

```

`\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).

```

1887 \newcommand*{\columnseparator}{%
1888     \smash{\rule[-0.2\baselineskip]{\columnrulewidth}{1.05\baselineskip}}}
1889 \newdimen\columnrulewidth
1890 \columnrulewidth=\z@
1891

```

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

```

1892 \newcommand*{\columnspostion}[1]{%
1893     \xdef\columns@position{#1}%
1894 }%
1895 \xdef\columns@position{R}%

```

`\beforecolumnseparator` `\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`.

```

1896 \newlength{\beforecolumnseparator}%
1897 \setlength{\beforecolumnseparator}{-2pt}%
1898
1899 \newlength{\aftercolumnseparator}%
1900 \setlength{\aftercolumnseparator}{-2pt}%
1901

```

```

setwidthliketwocolumns@L The \setwidth... macros are called in \beginnumbering in a not parallel
tpositionliketwocolumns@L typesetting, to fix the width of the lines to be vertically aligned with parallel
epositionliketwocolumns@L columns. They are also called at the beginning of a note's group, if some options
setwidthliketwocolumns@C are enabled. The \setposition... macros are called in \beginnumbering in a
tpositionliketwocolumns@C not parallel to fix the position of the lines. The \setnoteposition... macros
epositionliketwocolumns@C are called \xxxfootstart in a not parallel to fix the position of notes block.

setwidthliketwocolumns@R 1902 \newcommand{\setwidthliketwocolumns@L}{%
tpositionliketwocolumns@R 1903 % Temporary dimension, initially equal to the standard hsize, i.e. text width
epositionliketwocolumns@R 1904 % \begin{macrocode}
1905 \newdimen\temp%
1906 \temp=\hsize%

Hsize : Left + Right width
1907 \hsize=\Lcolwidth%
1908 \advance\hsize\Rcolwidth%

Now, calculating the remaining space
1909 \advance\temp-\hsize%

And multiply the hsize by 2/3 of this space
1910 \multiply\temp by 2%
1911 \divide\temp by 3%
1912 \advance\hsize\temp%
1913 }%
1914
1915 \newcommand{\setpositionliketwocolumns@L}{%
1916 \renewcommand{\ledrlfill}{\hfill}%
1917 }%
1918
1919 \newcommand{\setnotespositionliketwocolumns@L}{%
1920 }%
1921
1922

1923 \newcommand{\setwidthliketwocolumns@C}{%
1924 % Temporary dimension, initially equal to the standard hsize, i.e. text width

1925 \newdimen\temp%
1926 \temp=\hsize%
1927 % Hsize : Left + Right width

1928 \hsize=\Lcolwidth%
1929 \advance\hsize\Rcolwidth%
1930 % Now, calculating the remaining space
1931 \advance\temp-\hsize%

And multiply the hsize by 1/2 of this space
1932 \divide\temp by 2%
1933 \advance\hsize\temp%
1934 }%
1935
1936 \newcommand{\setpositionliketwocolumns@C}{%

```

```

1937 \doinsidelinehook{\hfill}%
1938 \renewcommand{\ledrlfill}{\hfill}%
1939 }%
1940
1941 \newcommand{\setnotespositionliketwocolumns@C}{%
1942 \newdimen\temp%
1943 \newdimen\tempa%
1944 \temp=\hsize%
1945 \tempa=\Lcolwidth%
1946 \advance\tempa\Rcolwidth%
1947 \advance\temp-\tempa%
1948 \divide\temp by 2%
1949 \leftskip=\temp%
1950 \rightskip=-\temp%
1951 }%
1952
1953 \newcommand{\setwidthliketwocolumns@R}{%
    Temporary dimension, initially equal to the standard hsize, i.e. text width
1954 \newdimen\temp%
1955 \temp=\hsize%
    Hsize : Left + Right width
1956 \hsize=\Lcolwidth%
1957 \advance\hsize\Rcolwidth%
    Now, calculating the remaining space
1958 \advance\temp-\hsize%
    And multiply the hsize by 2/3 of this space
1959 \multiply\temp by 2%
1960 \divide\temp by 3%
1961 \advance\hsize\temp%
1962 }%
1963
1964 \newcommand{\setpositionliketwocolumns@R}{%
1965 \doinsidelinehook{\hfill}%
1966 }%
1967
1968 \newcommand{\setnotespositionliketwocolumns@R}{%
1969 \newdimen\temp%
1970 \newdimen\tempa%
1971 \temp=\hsize%
1972 \tempa=\Lcolwidth%
1973 \advance\tempa\Rcolwidth%
1974 \advance\temp-\tempa%
1975 \divide\temp by 2%
1976 \leftskip=\temp%
1977 \rightskip=-\temp%
1978 }%
1979

```

27 Parallel pages

This is considerably more complicated than parallel columns.

```

\numpagelinesL Counts for the number of lines on a left or right page, and the smaller of the
\numpagelinesR number of lines on a pair of facing pages.
\l@dminpagelines 1980 \newcount\numpagelinesL
                  1981 \newcount\numpagelinesR
                  1982 \newcount\l@dminpagelines
                  1983

\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.

1984 \newcommand*{\Pages}{%
1985   \eledsection@correcting@skip=-2\baselineskip% line correcting for section titles.
1986   \setcounter{pstartL}{\value{pstartLold}}
1987   \setcounter{pstartR}{\value{pstartRold}}
1988   \parledgroup@notespacing@set@correction
1989   \typeout{}
1990   \typeout{***** PAGES *****}
1991   \ifnum\l@dnumpstartsL=\l@dnumpstartsR\else
1992     \led@err@BadLeftRightPstarts{\the\l@dnumpstartsL}{\the\l@dnumpstartsR}%
1993   \fi

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

1994   \cleartol@devenpage
1995   \begingroup
1996     \l@dzeropenalties
1997     \endgraf\global\num@lines=\prevgraf
1998     \global\num@linesR=\prevgraf
1999     \global\par@line=\z@
2000     \global\par@lineR=\z@
2001     \global\l@dpscL=\z@
2002     \global\l@dpscR=\z@
2003     \writtenlinesLfalse
2004     \writtenlinesRfalse

  Check if there are chunks to be processed.

2005     \check@pstarts
2006     \loop\if@pstarts

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

2007       \global\advance\l@dpscL \@ne
2008       \global\advance\l@dpscR \@ne

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

2009       \getlinesfromparlistL

```

```

2010 \getlinesfromparlistR
2011 \l@dcalc@maxoftwo{\@cs@linesinparL}{\@cs@linesinparR}%
2012 {\usernamecount{l@dmaxlinesinpar\the\l@dpscL}}%
2013 \check@pstarts
2014 \repeat

```

Zero the counts again, ready for the next bit.

```

2015 \global\l@dpscL=\z@
2016 \global\l@dpscR=\z@

```

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

```

2017 \getlinesfrompagelistL
2018 \getlinesfrompagelistR
2019 \l@dcalc@minoftwo{\@cs@linesonpageL}{\@cs@linesonpageR}%
2020 {\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.

```

2021 \check@pstarts
2022 \if@pstarts

```

Increment the chunk counts to get the first pair.

```

2023 \global\advance\l@dpscL \@ne
2024 \global\advance\l@dpscR \@ne

```

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

```

2025 \global\@donereallinesL=\z@
2026 \global\@donetotallinesL=\z@
2027 \global\@donereallinesR=\z@
2028 \global\@donetotallinesR=\z@

```

Start a loop over the boxes (chunks).

```

2029 \checkraw@text
2030 % \begingroup
2031 { \loop\ifaraw@text

```

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

```

2032 \checkpageL
2033 \l@duselanguage{\theledlanguageL}%
2034 %%% \begingroup
2035 { \loop\ifl@dsamepage%

```

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

```

2036 \ifdefstring{\@eledsectnotoc}{L}{\ledsectnotoc}{L}
2037 \csuse{before@pstartL@the\l@dpscL}
2038 \global\csundef{before@pstartL@the\l@dpscL}
2039 \do@lineL
2040 \xifinlist{\the\l@dpscL}{\eled@sections@}

```



```

2041      {\print@eledsectionL}%
2042      {}%
2043      \advance\numpagelinesL \@ne
2044      \ifshiftedpstarts
2045          \ifdim\ht\l@dleftbox>0pt\hb@xt@ \hsize{\ledstrutL\unhbox\l@dleftbox}\fi%
2046      \else%
2047          \parledgroup@correction@notespacing{L}
2048          \hb@xt@ \hsize{\ledstrutL\unhbox\l@dleftbox}%
2049      \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.

```

2050      \checkpageL%
2051      \get@nextboxL%
2052      \checkverseL
2053      \checkpbL
2054      \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.

```

2055      \ifl@dpagfull
2056          \@writelinesonpageL{\the\numpagelinesL}%
2057      \else
2058          \@writelinesonpageL{1000}%
2059      \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.

```

2060      \numpagelinesL \z@
2061      \parledgroup@correction@notespacing@init
2062      \clearl@dleftpage }%

```

Now do the same for the right text.

```

2063      \checkpageR
2064      \l@duselanguage{\theledlanguageR}%
2065      {
2066          \loop\ifl@dsamepage%
2067              \initnumbering@sectcountR
2068              \ifdefstring{\@eledsectnotoc}{R}{\ledsectnotoc}{}
2069              \csuse{before@pstartR@the\l@dpscR}
2070              \global\csundef{before@pstartR@the\l@dpscR}
2071              \do@lineR
2072              \xifinlist{\the\l@dpscR}{\eled@sectionsR@@}
2073              {\print@eledsectionR}%
2074              {}%
2075          \advance\numpagelinesR \@ne
2076          \ifshiftedpstarts
2077              \ifdim\ht\l@drighbox>0pt\hb@xt@ \hsize{\ledstrutR\unhbox\l@drighbox}\fi%
2078          \else%

```

```

2078          \parledgroup@correction@notespacing{R}
2079          \hb@xt@ \hsize{\ledstrutR\unhbox\l@drightbox}%
2080      \fi
2081      \checkpageR%
2082      \get@nextboxR%
2083      \checkverseR
2084      \checkpbr
2085      \repeat
2086      \ifl@dpagfull
2087          \@writelinesonpageR{\the\numpagelinesR}%
2088      \else
2089          \@writelinesonpageR{1000}%
2090      \fi
2091      \numpagelinesR=\z@
2092      \parledgroup@correction@notespacing@init

```

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

```

2093      \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.

```

2094      \checkraw@text
2095      \ifaraw@text
2096          \getlinesfrompagelistL
2097          \getlinesfrompagelistR
2098          \l@dcalc@minoftwo{\@cs@linesonpageL}{\@cs@linesonpageR}%
2099                      {\l@dminpagelines}%
2100      \fi
2101      \repeat}

```

We have now output the text from all the chunks.

```

2102      \fi

```

Make sure that there are no inserts hanging around.

```

2103      \flush@notes
2104      \flush@notesR
2105      \endgroup

```

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

```

2106      \global\l@dpscL=\z@
2107      \global\l@dpscR=\z@
2108      \global\l@dnumpestartsL=\z@
2109      \global\l@dnumpestartsR=\z@
2110      \global\instanzaLfalse
2111      \global\instanzaRfalse
2112      \ignorespaces}
2113

```

\ledstrutL Struts inserted into leftand right text lines.

\ledstrutR

```

2114 \newcommand*{\ledstrutL}{\strut}
2115 \newcommand*{\ledstrutR}{\strut}
2116

```

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

```

2117 \providecommand{\cleartoevenpage}[1][\@empty]{%
2118   \clearpage
2119   \ifodd\c@page\hbox{ }#1\clearpage\fi}
2120 \newcommand*{\cleartol@devenpage}{%
2121   \ifdim\pagetotal<\topskip% on an empty page
2122   \else
2123     \clearpage
2124   \fi
2125   \ifodd\c@page\hbox{ }\clearpage\fi}
2126 \newcommand*{\clearl@dleftpage}{%
2127   \clearpage
2128   \ifodd\c@page\else
2129     \led@err@LeftOnRightPage
2130     \hbox{ }%
2131     \cleardoublepage
2132   \fi}
2133 \newcommand*{\clearl@drightpage}{%
2134   \clearpage
2135   \ifodd\c@page
2136     \led@err@RightOnLeftPage
2137     \hbox{ }%
2138     \cleartoevenpage
2139   \fi}
2140

```

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

```

\cs@linesinparR 2141 \newcommand*{\getlinesfromparlistL}{%
2142   \ifx\linesinpar@listL\empty
2143     \gdef\cs@linesinparL{0}%
2144   \else
2145     \gl@p\linesinpar@listL\to\cs@linesinparL
2146   \fi}
2147 \newcommand*{\getlinesfromparlistR}{%
2148   \ifx\linesinpar@listR\empty
2149     \gdef\cs@linesinparR{0}%
2150   \else
2151     \gl@p\linesinpar@listR\to\cs@linesinparR
2152   \fi}

```

2153

`\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 2154 \newcommand*{\getlinesfrompagelistL}{%
2155   \ifx\linesonpage@listL\empty
2156     \gdef\cs@linesonpageL{1000}%
2157   \else
2158     \gl@p\linesonpage@listL\to\cs@linesonpageL
2159   \fi}
2160 \newcommand*{\getlinesfrompagelistR}{%
2161   \ifx\linesonpage@listR\empty
2162     \gdef\cs@linesonpageR{1000}%
2163   \else
2164     \gl@p\linesonpage@listR\to\cs@linesonpageR
2165   \fi}
2166
```

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

```

2167 \newcommand*{\@writelinesonpageL}[1]{%
2168   \edef\next{\write\linenum@out{\string\@lopL{#1}}}%
2169   \next}
2170 \newcommand*{\@writelinesonpageR}[1]{%
2171   \edef\next{\write\linenum@outR{\string\@lopR{#1}}}%
2172   \next}
2173
```

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

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

```

2174 \newcommand*{\l@dcalc@maxoftwo}[3]{%
2175   \ifnum #2>#1\relax
2176     #3=#2\relax
2177   \else
2178     #3=#1\relax
2179   \fi}
2180 \newcommand*{\l@dcalc@minoftwo}[3]{%
2181   \ifnum #2<#1\relax
2182     #3=#2\relax
2183   \else
2184     #3=#1\relax
2185   \fi}
2186
```

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

\ifl@dsamepage is set TRUE. If the page is spatially full then \ifl@dpagfull is set TRUE and \ifl@dsamepage is set FALSE. If it is not spatially full but the maximum number of lines have been output then both \ifl@dpagfull and \ifl@dsamepage are set FALSE.

```

2187 \newif\ifl@dsamepage
2188 \l@dsamepagetrue
2189 \newif\ifl@dpagfull
2190
2191 \newcommand*{\checkpageL}{%
2192   \l@dpagfulltrue
2193   \l@dsamepagetrue
2194   \check@goal
2195   \ifdim\pagetotal<\ledthegoal
2196     \ifnum\numpagelinesL<\l@dminpagelines
2197       \else
2198         \l@dsamepagefalse
2199         \l@dpagfullfalse
2200       \fi
2201     \else
2202       \l@dsamepagefalse
2203       \l@dpagfulltrue
2204     \fi}
2205 \newcommand*{\checkpageR}{%
2206   \l@dpagfulltrue
2207   \l@dsamepagetrue
2208   \check@goal
2209   \ifdim\pagetotal<\ledthegoal
2210     \ifnum\numpagelinesR<\l@dminpagelines
2211       \else
2212         \l@dsamepagefalse
2213         \l@dpagfullfalse
2214       \fi
2215     \else
2216       \l@dsamepagefalse
2217       \l@dpagfulltrue
2218     \fi}
2219

```

\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.

```

2220 \newcommand{\checkpbL}{
2221   \IfStrEq{\led@pb@setting}{after}{
2222     \xifinlistcs{\the\absline@num}{\l@prev@pb}{\l@dpagfulltrue\l@dsamepagefalse}{
2223     \xifinlistcs{\the\absline@num}{\l@prev@nopb}{\l@dpagfullfalse\l@dsamepagetrue}{
2224   }}
2225   \IfStrEq{\led@pb@setting}{before}{
2226     \numdef{\next@absline}{\the\absline@num+1}
2227     \xifinlistcs{\next@absline}{\l@prev@pb}{\l@dpagfulltrue\l@dsamepagefalse}{

```

```

2228     \xifinlistcs{\next@absline}{l@prev@nopb}{\l@dpagfullfalse\l@dsamepagetrue}{}
2229   }{}
2230 }
2231
2232 \newcommand{\checkpbR}{
2233   \IfStrEq{\led@pb@setting}{after}{
2234     \xifinlistcs{\the\absline@numR}{l@prev@pbR}{\l@dpagfulltrue\l@dsamepagefalse}{}
2235     \xifinlistcs{\the\absline@numR}{l@prev@nopbR}{\l@dpagfullfalse\l@dsamepagetrue}{}
2236   }{}
2237   \IfStrEq{\led@pb@setting}{before}{
2238     \numdef{\next@abslineR}{\the\absline@numR+1}
2239     \xifinlistcs{\next@abslineR}{l@prev@pbR}{\l@dpagfulltrue\l@dsamepagefalse}{}
2240     \xifinlistcs{\next@abslineR}{l@prev@nopbR}{\l@dpagfullfalse\l@dsamepagetrue}{}
2241   }{}
2242 }

```

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

```

2243 \newcommand{\checkverseL}{
2244   \ifinstanzaL
2245     \iflednopbinverse
2246       \ifinserthangingsymbol
2247         \numgdef{\prev@abslineverse}{\the\absline@num-1}
2248         \IfStrEq{\led@pb@setting}{after}{\lednopbnum{\prev@abslineverse}}{}
2249         \IfStrEq{\led@pb@setting}{before}{\ifnum\numpagelinesL<3\ledpbnum{\prev@abslineverse}}{}
2250       \fi
2251     \fi
2252   \fi
2253 }
2254 \newcommand{\checkverseR}{
2255   \ifinstanzaR
2256     \iflednopbinverse
2257       \ifinserthangingsymbolR
2258         \numgdef{\prev@abslineverse}{\the\absline@numR-1}
2259         \IfStrEq{\led@pb@setting}{after}{\lednopbnumR{\prev@abslineverse}}{}
2260         \IfStrEq{\led@pb@setting}{before}{\ifnum\numpagelinesR<3\ledpbnumR{\prev@abslineverse}}{}
2261       \fi
2262     \fi
2263   \fi
2264 }

```

`\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`.

```

2265 \newdimen\ledthegoal
2266 \ifshiftedpstarts
2267   \newcommand*{\goalfraction}{0.95}
2268 \else
2269   \newcommand*{\goalfraction}{0.9}

```

```

2270 \fi
2271
2272 \newcommand*{\check@goal}{%
2273   \ledthegoal=\goalfraction\pagegoal}
2274

```

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

```

\ifwrittenlinesL 2275 \newif\ifwrittenlinesL
2276 \newif\ifwrittenlinesR
2277

```

\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.

```

2278 \newcommand*{\get@nextboxL}{%
2279   \ifvbox\namebox{\l@dLcolrawbox\the\l@dpscl}% box is not empty

   The current box is not empty; do nothing.

2280   \else%                                     box is empty

   The box is empty. Check if enough lines (real and blank) have been output.
2281     \ifnum\usenamecount{\l@dmaxlinesinpar\the\l@dpscl}>\@donetotallinesL
2282       \parledgroup@notes@endL
2283     \else

   Sufficient lines have been output.
2284       \ifnum\usenamecount{\l@dmaxlinesinpar\the\l@dpscl}=\@donetotallinesL
2285         \parledgroup@notes@endL
2286       \fi
2287     \ifwrittenlinesL\else

   Write out the number of lines done, and set the boolean so this is only done once.
2288       \@writelinesinparL
2289       \writtenlinesLtrue
2290     \fi
2291     \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.

2292       \writtenlinesLfalse
2293       \ifbypstart@
2294         \ifnum\value{pstartL}<\value{pstartLold}
2295         \else
2296           \global\line@num=0
2297           \resetprevline@
2298         \fi
2299       \fi

2300 % Add the content of the optional argument of the previous \cs{pend}.
2301 %   \begin{macrocode}
2302   \csuse{after@pendL@\the\l@dpscl}%
2303   \global\csundef{after@pendL@\the\l@dpscl}%

```

```

2304 % \end{macrocode}
2305 % \begin{macrocode}
2306 \addtocounter{pstartL}{1}
2307 \global\pstartnumtrue
2308 \l@dcalc@maxoftwo{\the\usernamecount{l@dmmaxlinesinpar\the\l@dpscL}}%
2309 \the\@donetotallinesL}%
2310 \usernamecount{l@dmmaxlinesinpar\the\l@dpscL}}%
2311 \global\@donetotallinesL \z@
2312 \global\advance\l@dpscL \@ne

Add notes of parallel ledgroup.

2313 \parledgroup@notes@endL
2314 \parledgroup@correction@notespadding@final{L}
2315 \else
2316 % Add the content of the optional argument of the last \cs{pend}.
2317 % \begin{macrocode}
2318 \l@dpagefulltrue
2319 \l@dsamepagefalse%
2320 \csuse{after@pendL@the\l@dpscL}%
2321 \global\csundef{after@pendL@the\l@dpscL}%
2322 % \end{macrocode}
2323 % \begin{macrocode}
2324 \fi
2325 \fi
2326 \fi}

2327 \newcommand*{\get@nextboxR}{%
2328 \ifvbox\namebox{l@dRcolrawbox\the\l@dpscR}% box is not empty
2329 \else% box is empty
2330 \ifnum\usernamecount{l@dmmaxlinesinpar\the\l@dpscR}>\@donetotallinesR
2331 \parledgroup@notes@endR
2332 \else
2333 \ifnum\usernamecount{l@dmmaxlinesinpar\the\l@dpscR}=\@donetotallinesR
2334 \parledgroup@notes@endR
2335 \fi
2336 \ifwrittenlinesR\else
2337 \@writelinesinparR
2338 \writtenlinesRtrue
2339 \fi
2340 \ifnum\l@dnumpstartsR>\l@dpscR
2341 \writtenlinesRfalse
2342 \ifbypstart@R
2343 \ifnum\value{pstartR}<\value{pstartRold}
2344 \else
2345 \global\line@numR=0
2346 \resetprevline@
2347 \fi
2348 \fi
2349 \csuse{after@pendR@the\l@dpscR}%
2350 \global\csundef{after@pendR@the\l@dpscR}%
2351 \addtocounter{pstartR}{1}

```



```

2352      \global\pstartnumRtrue
2353      \l@dcalc@maxoftwo{\the\usernamecount{l@dmaxlinesinpar\the\l@dpscR}}%
2354      {\the\@donetotallinesR}%
2355      {\usernamecount{l@dmaxlinesinpar\the\l@dpscR}}}%
2356      \global\@donetotallinesR \z@
2357      \global\advance\l@dpscR \@ne
2358      \parledgroup@notes@endR
2359      \parledgroup@correction@notespadding@final{R}
2360  \else
2361      \l@dpagefulltrue%
2362      \l@dsamepagefalse%
2363      \csuse{after@pendR@\the\l@dpscR}%
2364      \global\csundef{after@pendR@\the\l@dpscR}%
2365  \fi
2366 \fi
2367 \fi}
2368

```

28 Sections' titles' commands

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

```

2369 \newcommand{\eledsectnotoc}[1]{\xdef\@eledsectnotoc{#1}}
2370 \eledsectnotoc{R}

```

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

```

2371 \newcommand{\eledsectmark}[1]{\xdef\@eledsectmark{#1}}
2372 \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`.

```

2373 \newskip\eledsection@correcting@skip

```

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

```

2374 \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).

```
2375 \def\l@prev@pbR{}
2376 \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.

```
2377 \newcommand{\ledpbR}{\write\linenum@outR{\string\led@pbR}}
2378 \newcommand{\ledpbnumR}[1]{\write\linenum@outR{\string\led@pbnumR{#1}}}
2379 \newcommand{\lednopbR}{\write\linenum@outR{\string\led@nopbR}}
2380 \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.

```
2381 \newcommand{\led@pbR}{\listcsxadd{\prev@pbR}{\the\absline@numR}}
2382 \newcommand{\led@pbnumR}[1]{\listcsxadd{\prev@pbR}{#1}}
2383 \newcommand{\led@nopbR}{\listcsxadd{\prev@nopbR}{\the\absline@numR}}
2384 \newcommand{\led@nopbnumR}[1]{\listcsxadd{\prev@nopbR}{#1}}
```

30 Parallel ledgroup

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

```
2385 \newmarks\parledgroup@
2386 \newmarks\parledgroup@series
2387 \newmarks\parledgroup@type
```

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

```
2388 \newcommand{\parledgroup@notes@startL}{%
2389   \ifnum\usenamecount{\l@dmxlinesinpar\the\l@dpscL}>0%
2390     \IfStrEq{\splitfirstmarks\parledgroup@type}{footnoteX}{\csuse{bhooknoteX@splitfirstmarks}}
2391     \IfStrEq{\splitfirstmarks\parledgroup@type}{Xfootnote}{\csuse{bhookXnote@splitfirstmarks}}
2392   \fi%
2393   \global\ledgroupnotesL@true%
2394   \insert@noterule@ledgroup{L}%
2395 }
2396 \newcommand{\parledgroup@notes@startR}{%
2397   \ifnum\usenamecount{\l@dmxlinesinpar\the\l@dpscR}>0%
2398     \IfStrEq{\splitfirstmarks\parledgroup@type}{footnoteX}{\csuse{bhooknoteX@splitfirstmarks}}
2399     \IfStrEq{\splitfirstmarks\parledgroup@type}{Xfootnote}{\csuse{bhookXnote@splitfirstmarks}}
2400   \fi%
2401   \global\ledgroupnotesR@true%
```

```

2402 \insert@noterule@ledgroup{R}%
2403 }

```

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

```

2404 \newcommand{\parledgroup@notes@endL}{%
2405   \global\ledgroupnotesL=false%
2406 }
2407 \newcommand{\parledgroup@notes@endR}{%
2408   \global\ledgroupnotesR=false%
2409 }

```

`\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`

```

2410 \newcommand{\insert@noterule@ledgroup}[1]{
2411   \IfStrEq{\splitbotmarks\parledgroup@}{begin}{%
2412     \IfStrEq{\splitbotmarks\parledgroup@type}{Xfootnote}{
2413       \csuse{ifledgroupnotes#1@}
2414       \vskip\skip\csuse{mp\splitbotmarks\parledgroup@series footins}
2415       \csuse{\splitbotmarks\parledgroup@series footnoterule}
2416       \fi
2417     }
2418   {}
2419   \IfStrEq{\splitbotmarks\parledgroup@type}{footnoteX}{
2420     \csuse{ifledgroupnotes#1@}
2421     \vskip\skip\csuse{mpfootins\splitbotmarks\parledgroup@series}
2422     \csuse{footnoterule\splitbotmarks\parledgroup@series}
2423     \fi
2424   }{}
2425 }
2426 {}
2427 }

```

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

```

2428 \newcommand{\parledgroupnotespacing}{}

```

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

```

2429 \dimdef{\parledgroup@notespacing@correction}{0pt}
2430 \newcommand{\parledgroup@notespacing@set@correction}{%
2431   {\notefontsetup\parledgroupnotespacing\dimgdef{\temp@spacing}{\baselineskip}}%
2432   \dimgdef{\parledgroup@notespacing@correction}{\baselineskip-\temp@spacing}%
2433 }

```

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

```

2434 \newcommand{\parledgroup@correction@notespacing@init}{
2435   \dimdef{\parledgroup@notespacing@correction@accumulated}{0pt}
2436   \dimdef{\parledgroup@notespacing@correction@modulo}{0pt}
2437 }
2438 \parledgroup@correction@notespacing@init

```

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

```

2439 \newcommand{\parledgroup@correction@notespacing@final}[1]{
2440   \ifparledgroup
2441     \vspace{\parledgroup@notespacing@correction@accumulated}
2442     \parledgroup@correction@notespacing@init%
2443     \ifstrequal{#1}{L}{
2444       \numdef{\@checking}{\the\l@dpscL-1}
2445     }{
2446       \numdef{\@checking}{\the\l@dpscR-1}
2447     }
2448     \dimdef{\@beforenotes@current@diff}{\csuse{\parledgroup@beforenotes@\@checking L}-\cs
2449     \ifstrequal{#1}{L}%
2450     {% Left
2451       \ifdingreater{\@beforenotes@current@diff}{0pt}{\vspace{-\@beforenotes@current@diff}
2452     }%
2453     {% Right
2454       \ifdingreater{\@beforenotes@current@diff}{0pt}{\vspace{\@beforenotes@current@diff}}
2455     }%
2456   \fi
2457 }

```

`\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.

```

2458 {}
2459 \newcommand{\parledgroup@correction@notespacing}[1]{%
2460   \csuse{ifledgroupnotes#1@}%
2461   \vspace{-\parledgroup@notespacing@correction}%
2462   \dimdef{\parledgroup@notespacing@correction@accumulated}{\parledgroup@notespacing@correction@accumulated}

```

```

2463     \dimdef{\parledgroup@notespacing@correction@modulo}{\parledgroup@notespacing@correction@modulo+
2464     \ifdimless{\parledgroup@notespacing@correction@modulo}{\baselineskip}{\advance\numpagelinesL
2465     \dimdef{\parledgroup@notespacing@correction@modulo}{\parledgroup@notespacing@correction@modulo-
2466     }% mean greater than equal
2467     \fi%
2468 }

```

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

```

2469 \dimdef\parledgroup@beforenotesL{0pt}
2470 \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.

```

2471 \newcommand{\parledgroup@beforenotes@save}[1]{
2472   \ifparledgroup
2473     \csdingdef{@parledgroup@beforenotes@the\csuse{1@dnumstarts#1}#1}{\csuse{parledgroup@beforenotes
2474     \csdingdef{parledgroup@beforenotes#1}{0pt}
2475   \fi
2476 }

```

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 `\setstanzaindents` command.

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

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

See the two following examples:

With standard `\hangingsymbol`:

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

With the modification of `\hangingsymbol`:

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

References

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

Index

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

Symbols	<code>\@adv</code>	367, 632, 633
<code>\&</code>	1574, 1575, 1579, 1596, 1617	<code>\@afterindentfalse</code> 762
<code>\@M</code>	1585	<code>\@arabic</code> 223, 224, 811, 814

\@astanza@line 1595, 1601, <u>1604</u>	\@nobreakefalse 820, 864, 1005
\@auxout 1408, 1420, 1682	\@nobreaktrue	. 818, 822, 862, 866, 1005
\@beforenotes@current@diff 2448, 2451, 2454	\@nopbtrue 1875, 1876, 1881, 1882
\@chapter 763	\@oldnbreak	818, 820, 862, 864, 915, 936
\@checking 2444, 2446, 2448	\@pbtrue 1873, 1874, 1879, 1880
\@cs@linesinparL 2011, <u>2141</u>	\@pend <u>574</u> , 1750
\@cs@linesinparR 2011, <u>2141</u>	\@pendR <u>574</u> , 1755
\@cs@linesonpageL	. . 2019, 2098, <u>2154</u>	\@pstartsfalse <u>1723</u>
\@cs@linesonpageR	. . 2019, 2098, <u>2154</u>	\@pstartstrue <u>1723</u>
\@currentlabel 854, 897	\@ref <u>546</u>
\@donereallinesL <u>954</u> , 983, 1750, 1752, 2025	\@ref@reg 572
\@donereallinesR <u>954</u> , 1056, 1755, 1757, 2027	\@schapter 763
\@donetotallinesL <u>954</u> , 984, 987, 2026, 2281, 2284, 2309, 2311	\@set <u>399</u> , 639, 640, 1827, 1831
\@donetotallinesR <u>954</u> , 1057, 1060, 2028, 2330, 2333, 2354, 2356	\@startstanza 775, 776, 798, 799
\@eled@sectioningfalse 1008	\@tag 669, 692, 1503
\@eled@sectioningtrue 1006	\@templ@d 1468, 1470
\@eledsectionL	. <u>1759</u> , 1792, 1808, 1809	\@templ@n 1469, 1470
\@eledsectionR	. <u>1759</u> , 1803, 1813, 1814	\@writelinesinparL	. . <u>1748</u> , 1823, 2288
\@eledsectmark	. 1003, 1788, 1799, 2371	\@writelinesinparR	. . <u>1748</u> , 1824, 2337
\@eledsectnotoc	1002, 2036, 2067, 2369	\@writelinesonpageL	. 2056, 2058, <u>2167</u>
\@insertR 1362–1364, 1377–1379	\@writelinesonpageR	. 2087, 2089, <u>2167</u>
\@l@dttempcnta 440, 442, 444, 445, 449, 451, 453, 454, 1114, 1153, 1154, 1156, 1158, 1161, 1162, 1179–1183, 1185, 1192, 1197, 1201, 1209, 1214, 1218, 1251, 1254, 1256, 1260	\@xloop 1375
\@l@dttempcntb 179, 181, 183, 1144, 1145, 1192, 1197, 1201, 1209, 1214, 1218, 1243, 1247, 1260, 1268–1270, 1272, 1293– 1295, 1297, 1314–1316, 1318, 1449, 1450, 1478–1480, 1482, 1645–1649, 1653–1656, 1660–1663	A	
\@lab 559, 1399, 1411, <u>1436</u>	\absline@num 363, 433, 447, 466, 1085, 1871, 1879, 1881, 2222, 2223, 2226, 2247
\@lock 971, 1094	\absline@numR 52, <u>240</u> , 291, 294, 297, 430, 438, 459, 478, 512, 540, 551, 1067, 1106, 1107, 1144, 1361, 1872, 1880, 1882, 2234, 2235, 2238, 2258, 2381, 2383
\@lockR	. 57, 311, 313, 315, 328, 474, 490, 491, 493, 494, 522, 523, 525, 1043, 1076, 1120, 1122, 1123, 1125, 1206, 1223, 1225, 1227	\actionlines@list 281, 284, 433, 447, 466
\@lopL <u>583</u> , 2168	\actionlines@listR <u>244</u> , 259, 273, 276, 430, 438, 459, 478, 512, 540, 1166, 1169
\@lopR <u>583</u> , 2171	\actions@list	. 285, 434, 454, 468, 470
\@nl <u>289</u> , 615, 617	\actions@listR <u>244</u> , 260, 277, 431, 445, 461, 463, 480, 489, 514, 521, 541, 1170
\@nl@reg 338	\add@inserts 994, 1000
\@nl@regR 289	\add@inserts@nextR <u>1350</u>
		\add@insertsR <u>1350</u>
		\add@penaltiesL 982, <u>1371</u>
		\add@penaltiesR 1055, <u>1371</u>
		\addtocontents 1683–1685
		\adddtocounter 917, 938, 1001, 1834, 1835, 2306, 2351
		\advancelabel@refs 1406, 1418

- \advanceline 631, 662
 - \affixline@num 978
 - \affixline@numR 1050, 1176
 - \affixpstart@numL 992, 1283
 - \affixpstart@numR 1283
 - \affixside@note 994, 1000
 - \affixside@noteR 1455
 - \aftercolumnseparator . . 4, 1855, 1896
 - \appto 1461, 1462
 - \araw@textfalse 1735
 - \araw@texttrue 1735
 - astanza (environment) 8, 1577
 - \at@begin@pairs 739, 744, 745
 - \AtBeginDocument . . . 1432, 1667, 1693
 - \AtBeginPairs 4, 744
- B**
- \ballast@count 1104, 1109
 - \bbl@main@language 1709, 1710
 - \bbl@set@language 1700, 1701
 - \beforecolumnseparator 4, 1849, 1852, 1858, 1896
 - \beginnumbering 7, 786, 825
 - \beginnumberingR . . . 43, 124, 786, 869
 - \bfseries 811, 814
 - \bypage@Rfalse 144, 159, 164
 - \bypage@Rtrue 144, 154
 - \bypstart@Rfalse 144, 155, 165
 - \bypstart@Rtrue 144, 160
- C**
- \c@ballast 1109
 - \c@chapter 105
 - \c@chapterR 105
 - \c@firstlinenumR 188, 1249
 - \c@firstsublinenumR 192, 1244
 - \c@linenumincrementR 188, 1249
 - \c@page 615, 617, 2119, 2125, 2128, 2135
 - \c@pstart 1421
 - \c@pstartL 811
 - \c@pstartR 814, 1409
 - \c@section 106
 - \c@sectionR 106
 - \c@sublinenumincrementR . . 192, 1244
 - \c@subsection 107
 - \c@subsectionR 107
 - \c@subsubsection 108
 - \c@subsubsectionR 108
 - \ch@ck@l@ckR 1176
 - \ch@cksub@l@ckR 1176
 - \ch@cksub@lockR 1245
 - \chapter 748, 749, 758
 - \chapterinpages 735, 749, 760
 - \chardef 1574
 - \check@goal 2194, 2208, 2265
 - \check@pstarts 1723, 1776, 1825, 2005, 2013, 2021
 - \checkpageL 2032, 2050, 2187
 - \checkpageR 2063, 2081, 2187
 - \checkpb@columns . . . 1821, 1863, 1867
 - \checkpbl 2053, 2220
 - \checkpbr 2084, 2220
 - \checkraw@text 1735, 1782, 1818, 2029, 2094
 - \checkverseL 1819, 2052, 2243
 - \checkverseR 1820, 2083, 2243
 - \cleardoublepage 2131
 - \clearl@dleftpage 2062, 2117
 - \clearl@drighpage 2093, 2117
 - \cleartoevenpage 2117
 - \cleartol@devenpage 1994, 2117
 - \closeout 96, 596, 602, 605, 609
 - \columnrulewidth 4, 1887
 - \Columns 3, 1761
 - \columns@position . . 1806, 1816, 1892
 - \columnseparator 4, 1854, 1887
 - \columnspostion 4, 1892
 - \correcthangingL 996, 1555
 - \correcthangingR 1555
 - \countLline 949, 960
 - \countRline 949, 1032
 - \critext 667
 - \cs 1864, 2300, 2316
 - \csdimgdef 2473, 2474
 - \csexpandonce . . 1505, 1509, 1525, 1536
 - \csgdef 858, 901, 922, 943
 - \csundef 1009, 1791, 1802, 2038, 2069, 2303, 2321, 2350, 2364
 - \csuse 1007, 1524, 1535, 1789, 1800, 2037, 2068, 2302, 2320, 2349, 2363, 2390, 2391, 2398, 2399, 2413–2415, 2420–2422, 2448, 2460, 2473
- D**
- \DeclareOption 8–11
 - \def@tempb 162
 - \dimdef 2429, 2435, 2436, 2448, 2462, 2463, 2465, 2469, 2470
 - \dimen 618, 619, 623–625, 629

- `\dingdef` 2431, 2432
`\divide` 1181, 1911, 1932, 1948, 1960, 1975
`\do@actions` 1086
`\do@actions@fixedcodeR` 1113
`\do@actions@nextR` 1113
`\do@actionsR` 1068, 1113
`\do@ballast` 1087
`\do@ballastR` 1069, 1104
`\do@insidelineLhook` .. 996, 1022, 1025
`\do@insidelineRhook` 1023, 1025
`\do@lineL` 959, 1785, 2039
`\do@lineLhook` 964, 1020, 1025
`\do@lineR` 1030, 1796, 2070
`\do@lineRhook` 1021, 1025, 1036
`\do@lockoff` 509
`\do@lockoffL` 533
`\do@lockoffR` 509
`\do@lockon` 474
`\do@lockonL` 506
`\do@lockonR` 474
`\doinsidelinehook` 1937, 1965
`\doinsidelineLhook` 1020
`\doinsidelineRhook` 1020
`\dolineLhook` 1020
`\dolineRhook` 1020
`\dolistloop` 1466, 1486, 1493
`\dummy@ref` 555
- E**
- `\edfont@info` 718, 721, 727, 730
`\edlabel` 1397
`\edtext` 690
`\eled@sectioningR@out` .. 71, 96, 2374
`\eled@sections@@` 979, 1005, 1786, 2040
`\eled@sectionsR@@` 68, 1051, 1797, 2071
`\eledpar@error` 22, 31, 34, 36
`\eledsection@correcting@skip` ...
..... 1017, 1762, 1985, 2373
`\eledsectmark` 12, 2371
`\eledsectnotoc` 12, 2369
`\empty` .. 81, 84, 273, 281, 680, 703,
716, 725, 834, 878, 1166, 1248,
1256, 1352–1354, 1365, 1376,
1400, 1412, 2142, 2148, 2155, 2161
`\end@lemmas` 680, 681, 703, 704
`\endashchar` 1387
`\endgraf` 911, 932, 1770, 1997
`\endline@num` 562, 568
`\endlock` 651, 1583, 1592, 1597
`\endnumbering` 7, 74, 128, 787
`\endnumberingR` 46, 74, 111, 123, 136, 787
`\endpage@num` 561, 568
`\endstanzaextra` 1599
`\endsub` 618
`\endsubline@num` 563, 569
`\enlargethispage` 1884
environments:
 `astanza` 8, 1577
 `Leftside` 5, 767
 `pages` 5, 735
 `pairs` 3, 735
 `Rightside` 5, 784
`\extensionchars` 63, 117, 133, 141
- F**
- `\f@x@l@cksR` 1176
`\first@linenum@out@Rfalse` .. 591, 597
`\first@linenum@out@Rtrue` 591
`\firstlinenum` 5, 197
`\firstlinenum*` 5, 197
`\firstsublinenum` 5, 197
`\firstsublinenum*` 5, 197
`\fix@page` 334, 341
`\flag@end`
 . 618, 676, 686, 687, 699, 709, 710
`\flag@start`
 . 618, 675, 676, 687, 698, 699, 710
`\flush@notes` 1837, 2103
`\flush@notesR` 1374, 1838, 2104
`\footnotelang@lua` 1519, 1530
`\footnotelang@poly` 1523, 1534
`\fullstop` .. 236, 1384, 1386, 1388, 1390
- G**
- `\get@linelistfile` 269
`\get@nextboxL` 2051, 2278
`\get@nextboxR` 2082, 2278
`\getline@numL` 970, 1084
`\getline@numR` 1042, 1066
`\getlinesfrompagelistL`
..... 2017, 2096, 2154
`\getlinesfrompagelistR`
..... 2018, 2097, 2154
`\getlinesfromparlistL` ... 2009, 2141
`\getlinesfromparlistR` ... 2010, 2141
`\gl@p` 276, 277,
284, 285, 681, 704, 720, 729,
1169, 1170, 1358, 1362, 1377,
1403, 1415, 2145, 2151, 2158, 2164
`\goalfraction` 5, 2265

H

\hangingsymbol 10, 1546, 1552
 \hb@xt@ 977, 986, 996, 1049,
 1059, 1805, 2045, 2048, 2076, 2079
 \hsize 852, 895, 1805, 1906–
 1909, 1912, 1926, 1928, 1929,
 1931, 1933, 1944, 1955–1958,
 1961, 1971, 2045, 2048, 2076, 2079

I

\if@filesw 1681
 \if@firstcolumn 1262, 1287, 1308, 1472
 \if@ledgroup 601
 \if@nobreak 817, 861
 \if@noeled@sec 69, 95
 \if@nopb 1869, 1884
 \if@pb 1868, 1885
 \if@pstarts 1723, 1777, 2006, 2022
 \if@RTL 676, 687, 699, 710, 1010
 \ifaraw@text ... 1735, 1783, 2031, 2095
 \ifautopar 844, 888
 \ifbypage@ 357
 \ifbypage@R 144, 347, 1148
 \ifbypstart@ 576, 1826, 2293
 \ifbypstart@R ... 144, 580, 1830, 2342
 \ifdefstring 1002, 1003,
 1788, 1799, 1806, 1816, 2036, 2067
 \ifdim ... 619, 623, 625, 629, 1849,
 1855, 2045, 2076, 2121, 2195, 2209
 \ifdimgreater 2451, 2454
 \ifdimless 2464
 \iffirst@linenum@out@R 591, 595
 \ifhbox 1808, 1813
 \ifinserthangingsymbol
 1544, 1558, 2246
 \ifinserthangingsymbolR
 1542, 1550, 1568, 2257
 \ifinstanzaL 765, 765, 1545, 1557, 2244
 \ifinstanzaR 765, 766, 1551, 1567, 2255
 \ifl@d@dash 1387
 \ifl@d@elin 1389, 1390
 \ifl@d@esl 1390
 \ifl@d@pnum 1384, 1388
 \ifl@d@ssub 1386
 \ifl@dpagefull 2055, 2086, 2187
 \ifl@dpadding 13, 1556, 1566
 \ifl@dpairing 13, 78
 \ifl@dsamelang 1678
 \ifl@dsamepage 2035, 2065, 2187
 \ifl@dskipnumber 1239

\ifl@dusedbabel 1676
 \iflabelpstart 854, 897
 \ifledgroupnotesL@ 1088
 \ifledgroupnotesR@ 1070, 1238
 \iflednopbinverse 2245, 2256
 \ifledplinenum 1385
 \ifledRcol 13, 180,
 202, 206, 210, 214, 256, 271,
 335, 344, 369, 383, 400, 417,
 429, 437, 458, 503, 530, 539,
 548, 620, 626, 632, 639, 647,
 652, 656, 661, 671, 694, 715,
 1398, 1437, 1504, 1517, 1704, 1716
 \ifluatex 1518, 1529
 \ifnoteschanged@ 88
 \ifnumberedpar@
 ... 827, 871, 907, 928, 1502, 1516
 \ifnumbering 149, 823, 904
 \ifnumberingR ... 44, 75, 113, 867, 925
 \ifnumberline 1070, 1088, 1238
 \ifnumberpstart ... 845, 889, 916, 937
 \ifnumequal 1459
 \ifnumgreater 1467, 1487, 1494
 \ifodd 1272, 1297,
 1318, 1482, 2119, 2125, 2128, 2135
 \ifparledgroup 2440, 2472
 \ifpst@rtedL 39, 831
 \ifpst@rtedR 39, 875
 \ifpstartnum 1328, 1333
 \ifpstartnumR 1283
 \ifshiftedpstarts 5, 2044, 2075, 2266
 \ifsidepstartnum 846, 890, 1285, 1306
 \ifstrempy 856, 899, 920, 941
 \IfStrEq 968, 1040, 1870,
 1878, 2221, 2225, 2233, 2237,
 2248, 2249, 2259, 2260, 2390,
 2391, 2398, 2399, 2411, 2412, 2419
 \ifstrequal 2443, 2449
 \ifsublines@ 234,
 323, 368, 401, 408, 439, 448,
 460, 467, 479, 513, 567, 569,
 1071, 1089, 1155, 1242, 1439, 1443
 \ifvbox 961, 1033, 1739, 1742, 2279, 2328
 \ifvnode 1405, 1417
 \ifwidthliketwocolumns 11
 \ifwrittenlinesL 2275, 2287
 \ifwrittenlinesR 2276, 2336
 \initnumbering@sectcmd 738, 752
 \initnumbering@sectcountR
 67, 100, 120, 2066

- \InputIfFileExists 70
 - \insert@count [545](#), 672, 695, 1511, 1538
 - \insert@countR [546](#), 671, 694, 1507, 1527
 - \insert@noterule@ledgroup
..... 2394, 2402, [2410](#)
 - \inserthangingsymbolfalse 974
 - \inserthangingsymbolL 996, [1542](#)
 - \inserthangingsymbolR [1542](#)
 - \inserthangingsymbolRfalse 1046
 - \inserthangingsymbolRtrue 1044
 - \inserthangingsymboltrue 972
 - \insertlines@listR
.... 81, [244](#), 258, 551, 1354, 1358
 - \inserts@list 833, 1510, 1537
 - \inserts@listR 877, [1349](#),
1352, 1362, 1376, 1377, 1506, 1526
 - \instanzaLfalse 1845, 2110
 - \instanzaLtrue 776
 - \instanzaRfalse 1846, 2111
 - \instanzaRtrue 799
 - \interlinepenalty 1585
 - \itemcount@ 1457, 1459,
1464, 1467, 1485, 1487, 1492, 1494
- L**
- \l@dd@nums 718, 721, 727, 730
 - \l@d@set [416](#), 647, 648
 - \l@d@bbl@set@language [1678](#), 1701
 - \l@dbfnote [1501](#)
 - \l@dc@maxchunks 839, 841,
883, 885, [1635](#), 1645, 1653, 1660
 - \l@dcalc@maxoftwo
..... 2011, [2174](#), 2308, 2353
 - \l@dcalc@minoftwo .. 2019, 2098, [2174](#)
 - \l@dcalcnun [1176](#)
 - \l@dchecklang [1678](#)
 - \l@dchset@num 290, 293, [416](#)
 - \l@dcsnotetext 1466, 1469
 - \l@dcsnotetext@l 1469, 1486
 - \l@dcsnotetext@r 1469, 1493
 - \l@demptyd@ta 965, 1037
 - \l@dend@stuff 64, 118, 134, 142
 - \l@dgetline@margin 178
 - \l@dgetsidenote@margin 1448
 - \l@dld@ta 993,
1263, 1275, 1288, 1300, 1309, 1321
 - \l@dleftbox
.. [946](#), 976, 986, 1807, 2045, 2048
 - \l@dlinenumR [226](#)
 - \l@dlsn@te 995
 - \l@dmake@labels 1421
 - \l@dmake@labelsR 1409, [1426](#)
 - \l@dminpagelines
.... [1980](#), 2020, 2099, 2196, 2210
 - \l@dnumpsstartsL
..... 838, 839, 841, 843, 858,
922, [1639](#), 1671, 1726, 1765,
1766, 1842, 1991, 1992, 2108, 2291
 - \l@dnumpsstartsR
.. 48, 882, 883, 885, 887, 901,
943, [1639](#), 1672, 1729, 1765,
1766, 1843, 1991, 1992, 2109, 2340
 - \l@doldbbl@set@language 1700
 - \l@doldselectlanguage 1699, 1703, 1708
 - \l@dpagefullfalse
.... [2187](#), 2223, 2228, 2235, 2240
 - \l@dpagefulltrue [2187](#),
2222, 2227, 2234, 2239, 2318, 2361
 - \l@dpaddingfalse 15, 737, 757
 - \l@dpaddingtrue 751
 - \l@dpairingfalse 13, 741, 756
 - \l@dpairingtrue 736, 750
 - \l@dpscL 961, 966,
979, 1004, 1007, 1009, 1641,
1673, 1726, 1739, 1774, 1780,
1786, 1789, 1791, 1840, 2001,
2007, 2012, 2015, 2023, 2037,
2038, 2040, 2106, 2279, 2281,
2284, 2291, 2302, 2303, 2308,
2310, 2312, 2320, 2321, 2389, 2444
 - \l@dpscR 1033, 1038, 1051,
1642, 1674, 1729, 1742, 1775,
1781, 1797, 1800, 1802, 1841,
2002, 2008, 2016, 2024, 2068,
2069, 2071, 2107, 2328, 2330,
2333, 2340, 2349, 2350, 2353,
2355, 2357, 2363, 2364, 2397, 2446
 - \l@d@rd@ta 996,
1265, 1273, 1290, 1298, 1311, 1319
 - \l@d@rightbox
946, 1048, 1059, 1812, 2076, 2079
 - \l@d@rsn@te 997
 - \l@dsamepagefalse [2187](#),
2222, 2227, 2234, 2239, 2319, 2362
 - \l@dsamepagetrue
.... [2187](#), 2223, 2228, 2235, 2240
 - \l@dsetupmaxlinecounts .. [1652](#), 1669
 - \l@dsetuprawboxes [1644](#), 1668
 - \l@dskipnumberfalse 1240
 - \l@dskipnumbertrue 1136

- \l@dunhbox@line 996, 1012, 1015
- \l@dusedbabelfalse 1676, 1696
- \l@dusedbabeltrue 1676, 1698
- \l@duselanguage
 - 1688, 1784, 1795, 2033, 2064
- \l@dzeromaxlinecounts . . . 1652, 1670
- \l@dzeropenalties 910, 931, 1769, 1996
- \l@prev@nopbR 55, 2376
- \l@prev@pbR 54, 2375
- \l@pscL 1641
- \l@pscR 1641
- \label@refs
 - 1401, 1403, 1409, 1413, 1415, 1421
- \labelref@list 1412, 1415, 1444
- \labelref@listR 1395, 1400, 1403, 1440
- \languagename . . 1679, 1680, 1682–1685
- \last@page@num 355, 361
- \last@page@numR 341
- \lastbox 969, 1041
- \lastskip 618, 624
- \Lcolwidth
 - . . . 4, 5, 17, 753, 852, 977, 986,
 - 1792, 1907, 1928, 1945, 1956, 1972
- \led@err@BadLeftRightPstarts . . .
 - 30, 1766, 1992
- \led@err@LeftOnRightPage . . . 33, 2129
- \led@err@LineationInNumbered . . . 150
- \led@err@ManyLeftnotes 1487
- \led@err@ManyRightnotes 1494
- \led@err@ManySidenotes 1467
- \led@err@NumberingNotStarted . . . 92
- \led@err@NumberingShouldHaveStarted
 - 122
- \led@err@NumberingStarted 45
- \led@err@PendNoPstart 908, 929
- \led@err@PendNotNumbered . . . 905, 926
- \led@err@PstartInPstart . . . 828, 872
- \led@err@PstartNotNumbered . . 824, 868
- \led@err@RightOnLeftPage . . . 33, 2136
- \led@err@TooManyPstarts
 - 25, 27, 840, 884
- \led@mess@NotesChanged 89
- \led@mess@SectionContinued
 - 116, 132, 140
- \led@nopbnumR 2380, 2381
- \led@nopbR 2379, 2381
- \led@pb@setting
 - 1870, 1878, 2221, 2225,
 - 2233, 2237, 2248, 2249, 2259, 2260
- \led@pbnumR 2378, 2381
- \led@pbR 2377, 2381
- \led@warn@BadAction 1138
- \led@warn@BadAdvancelineLine 386, 392
- \led@warn@BadAdvancelineSubline .
 - 372, 378
- \led@warn@BadLineation 167
- \led@warn@BadSetline 637
- \led@warn@BadSetlinenum 645
- \led@warn@DuplicateLabel 1428
- \ledgroupnotesL@false 2405
- \ledgroupnotesL@true 2393
- \ledgroupnotesR@false 2408
- \ledgroupnotesR@true 2401
- \ledllfill 996
- \lednopb 795
- \lednopbnum 2248, 2377
- \lednopbnumR 2259, 2377
- \lednopbR 795, 2379
- \ledpb 794
- \ledpbnum 2249
- \ledpbnumR 2260, 2377
- \ledpbR 794, 2377
- \ledRcol@false 1062
- \ledRcol@true 1031
- \ledRcolfalse 16, 768, 801
- \ledRcoltrue 785
- \ledrlfill 996, 1916, 1938
- \ledsavedprintlines 8, 1382
- \ledsectnomark 1003
- \ledsectnotoc 1002, 2036, 2067
- \ledstrutL 2045, 2048, 2114
- \ledstrutR 2076, 2079, 2114
- \ledthegoal 2195, 2209, 2265
- \leftlinenumR 226, 1263, 1275
- \leftpstartnumL 1283
- \leftpstartnumR 1283
- Leftside (environment) 5, 767
- \Leftsidehook 774, 779
- \Leftsidehookend 778, 779
- \line@list 725, 729
- \line@list@stuff 133
- \line@list@stuffR . . . 63, 117, 141, 593
- \line@listR . . . 84, 244, 257, 569, 716, 720
- \line@margin 183, 1293
- \line@marginR 176, 1268, 1314
- \line@num . . . 358, 390, 391, 393, 411,
- 422, 423, 451, 576, 1095, 1442, 2296
- \line@numR 56,
- 233, 240, 295, 329, 348, 384,
- 385, 387, 404, 418, 419, 442,

562, 566, 580, 1077, 1149, 1158,
1247, 1249, 1251, 1252, 1438, 2345
`\lineation` 172, 173, 796
`\lineation*` 6, 172
`\lineationR` 6, 148, 174, 796
`\linenum@out`
 . 615, 621, 627, 633, 640, 648,
 653, 657, 1411, 1750, 1827, 2168
`\linenum@outR`
 . 590, 596, 598, 602, 603, 605,
 606, 609, 610, 617, 620, 626,
 632, 639, 647, 652, 656, 661,
 1399, 1755, 1831, 2171, 2377–2380
`\linenumberlist` 1248, 1252
`\linenumincrement` 5, 197
`\linenumincrement*` 5, 197
`\linenummargin` 176
`\linenumr@p` 1385, 1389, 1438, 1442
`\linenumrepR` 223, 233
`\linenumsep`
 . 228, 230, 1330, 1333, 1342, 1345
`\linesinpar@listL`
 249, 265, 577, 2142, 2145
`\linesinpar@listR`
 249, 261, 581, 2148, 2151
`\linesonpage@listL` 266, 585, 2155, 2158
`\linesonpage@listR` 262, 588, 2161, 2164
`\list@clear`
 . 257–262, 265, 266, 268, 833, 877
`\list@clearing@reg` 264
`\list@create`
 ... 244–247, 249–251, 1349, 1395
`\listcsxadd` 363, 2381–2384
`\lock@disp` 1208, 1212, 1217
`\lock@off` 500, 501, 509, 656, 657
`\lock@on` 652, 653

M

`\managestanza@modulo` 1611
`\maxchunks` 3, 1635
`\maxlinesinpar@list` 249, 268
`\memorydump` 7, 773, 790
`\memorydumpL` 127, 773
`\memorydumpR` 127, 790
`\message` 62
`\multiply` 1182, 1910, 1959

N

`\n@num` 537, 661
`\n@num@reg` 543

`\namebox` 961, 966, 1033,
 1038, 1619, 1739, 1742, 2279, 2328
`\NeedsTeXFormat` 2
`\new@line` 1012, 1015
`\new@lineL` 614, 996
`\new@lineR` 616
`\newbox` 806, 946, 947, 1620
`\newcommandx` 816, 860, 903, 924
`\newcounter` 100–103, 188,
 190, 192, 194, 809, 810, 812, 813
`\newif` 5,
 14, 40, 144, 145, 591, 765, 766,
 1337, 1542, 1676, 1723, 1735,
 1868, 1869, 2187, 2189, 2275, 2276
`\newlength` 1896, 1899
`\newmarks` 2385–2387
`\newnamebox` 1619, 1647, 1648
`\newnamecount` 1630, 1655
`\newsavebox` 1759, 1760
`\newwrite` 590, 2374
`\next@absline`
 1871, 1873, 1875, 2226–2228
`\next@abslineR`
 1872, 1874, 1876, 2238–2240
`\next@action` 285
`\next@actionline` 282, 284
`\next@actionlineR`
 . 274, 276, 1107, 1145, 1167, 1169
`\next@actionR` 277, 1108,
 1146, 1147, 1152, 1153, 1161, 1170
`\next@insert` 834
`\next@insertR`
 878, 1353, 1356, 1358, 1361, 1365
`\next@page@num` 362, 434
`\next@page@numR` 60, 298, 300, 352, 431
`\no@expands` 669, 692
`\noindent` 858, 901, 922, 943
`\normal@page@breakR` 53
`\normal@pars` 77, 837, 881
`\normalbfnoteX` 1515
`\notefontsetup` 2431
`\noteschanged@true`
 82, 85, 717, 726, 1355
`\notesXwidthliketwocolumns` 4
`\num@lines` 911, 1770, 1997
`\num@linesR` 805, 932, 1771, 1998
`\numberedpar@true` 853, 896
`\numberingRfalse` 76
`\numberingRtrue` 50, 111, 137
`\numberingtrue` 129

- \numberpstartfalse 8
- \numberpstarttrue 8
- \numdef 1004, 1485, 1492,
1871, 1872, 2226, 2238, 2444, 2446
- \numgdef 1457, 1464, 2247, 2258
- \numlabfont 233
- \numpagelinesL 1980,
2043, 2056, 2060, 2196, 2249, 2464
- \numpagelinesR
1980, 2074, 2087, 2091, 2210, 2260
- O**
- \old@otherlanguage 1713
- \old@startstanza .. 775, 776, 798, 799
- \oldchapter 748, 758
- \one@line ... 966, 969, 996, 1012, 1015
- \one@lineR 805, 1038, 1041
- \openout 71, 598, 603, 606, 610
- \otherlanguage 1713, 1714
- P**
- \p@pstartL 855
- \p@pstartR 898
- \PackageError 22
- \page@action 299, 428, 556
- \page@num 280, 360, 1295
- \page@numR 253, 272, 350,
561, 566, 1147, 1270, 1316, 1480
- \pagebreak 1885
- \pagegoal 2273
- \Pages 5, 1984
- pages (environment) 5, 735
- \pagetotal 2121, 2195, 2209
- pairs (environment) 3, 735
- \paperwidth 1011, 1014
- \par@line 912, 1772, 1999
- \par@lineR 805, 933, 1773, 2000
- \parbox 1792, 1803
- \parledgroup@ .. 968, 1040, 2385, 2411
- \parledgroup@beforenotes@save ..
..... 919, 940, 2471
- \parledgroup@beforenotesL 2469
- \parledgroup@beforenotesR 2469
- \parledgroup@correction@notespacing
..... 2047, 2078, 2458
- \parledgroup@correction@notespacing@fini
..... 2314, 2359, 2439
- \parledgroup@correction@notespacing@ini
..... 2061, 2092, 2434, 2442
- \parledgroup@notes@endL
..... 2282, 2285, 2313, 2404
- \parledgroup@notes@endR
..... 2331, 2334, 2358, 2407
- \parledgroup@notes@startL
..... 968, 2388, 2404
- \parledgroup@notes@startR
..... 1040, 2388, 2404
- \parledgroup@notespacing@correction
..... 2429, 2461–2463
- \parledgroup@notespacing@correction@accumulated
..... 2435, 2441, 2462
- \parledgroup@notespacing@correction@modulo
..... 2436, 2463–2465
- \parledgroup@notespacing@set@correction
..... 1988, 2429
- \parledgroup@series
..... 2386, 2390, 2391,
2398, 2399, 2414, 2415, 2421, 2422
- \parledgroup@type 2387,
2390, 2391, 2398, 2399, 2412, 2419
- \parledgroupnotespacing .. 2428, 2431
- \parledgroupseries@ 2385
- \parledgrouptrue 10
- \parledgrouptype@ 2385
- \pausenumbering 788
- \pausenumberingR 110, 788
- \pend 6, 772, 793, 829, 1598
- \pendL 772, 903
- \pendR 793, 873, 924
- \prev@abslineverse
..... 2247–2249, 2258–2260
- \prev@nopbR 2375
- \prev@pbR 2375
- \prevgraf
..... 911, 932, 1770, 1771, 1997, 1998
- \print@columnseparator .. 1811, 1848
- \print@eledsectionL
..... 991, 999, 1792, 2041
- \print@eledsectionR .. 1066, 1803, 2072
- \print@lineL 981, 991
- \print@lineR 1053, 1066
- \printlines 1393
- \printlinesR 8, 1382
- \ProcessOptions 12
- \protected@csxdef 1524, 1535
- \protected@edef 854, 897
- \protected@write ... 1408, 1420, 1682
- \ProvidesPackage 3
- \pst@rtedLfalse 39

- `\pst@rtedLtrue` 130, 835
`\pst@rtedRfalse` 41, 49, 79
`\pst@rtedRtrue` 114, 138, 879
`\pstart` 6, 28, 32, 770, 792, 1600
`\pstartL` 770, 808
`\pstartnumfalse` 1330, 1335
`\pstartnumRfalse` 1342, 1347
`\pstartnumRtrue` 1338, 1779, 2352
`\pstartnumtrue` 1778, 2307
`\pstartR` 792, 808
- R**
- `\Rcolwidth`
 4, 5, 17, 754, 895, 1049, 1059,
 1803, 1908, 1929, 1946, 1957, 1973
`\read@linelist` 255, 594
`\rem@inder` 1252, 1254–1256
`\resetprevline@` 1828, 1832, 2297, 2346
`\resumenumbering` 789
`\resumenumberingR` 110, 789
`\rightlinenumR` 226, 1265, 1273
`\rightpstartnumL` 1283
`\rightpstartnumR` 1283
Rightside (environment) 5, 784
`\Rightsidehook` 779, 797
`\Rightsidehookend` 779, 802
`\rlap` 1265, 1273, 1290, 1298, 1311, 1319
`\Rlineflag` 8, 221, 233, 1385, 1389, 1430
`\rule` 1888
- S**
- `\savebox` 1792, 1803
`\sc@n@list` 1253, 1255
`\secdef` 763
`\section@num` 131–133
`\section@numR` 37,
 51, 62, 63, 70, 71, 115–117, 139–141
`\select@language` 1680, 1682–1685, 1719
`\selectlanguage` 1688
`\set@line` 670, 693, 714
`\set@line@action`
 292, 397, 406, 413, 436, 558
`\setl@dlp@rbox` 1473, 1488, 1490
`\setl@drp@rbox` 1475, 1483, 1495
`\setline` 635
`\setlinenum` 643
`\setnamebox` 843, 887, 1619
`\setnotepositionliketwocolumns@C`
 1902
`\setnotepositionliketwocolumns@L`
 1902
`\setnotepositionliketwocolumns@R`
 1902
`\setnotespositionliketwocolumns@C`
 1941
`\setnotespositionliketwocolumns@L`
 1919
`\setnotespositionliketwocolumns@R`
 1968
`\setpositionliketwocolumns@C` ...
 1902, 1936
`\setpositionliketwocolumns@L` ...
 1902, 1915
`\setpositionliketwocolumns@R` ...
 1902, 1964
`\setprintlines` 1383
`\setwidthliketwocolumns@C` 1902, 1923
`\setwidthliketwocolumns@L` 1902, 1902
`\setwidthliketwocolumns@R` 1902, 1953
`\shiftedpstartsfalse` 7
`\shiftedpstartstrue` 6, 8, 9
`\shiftedversesfalse` 7
`\shiftedversestrue` 6
`\showlemma` 679, 702
`\sidenote@margin` 1450, 1453
`\sidenote@marginR` 1447, 1478
`\sidenotecontent@`
 . 1456, 1461, 1462, 1473, 1475,
 1483, 1484, 1488, 1490, 1491, 1495
`\sidenotemargin` 1447
`\sidenotemargin*` 1447
`\sidenotesep` 1462
`\skip` 2414, 2421
`\skip@lockoff` 501, 509
`\skipnumbering` 8, 660
`\skipnumbering@reg` 664
`\smash` 1888
`\splitbotmarks` 2411,
 2412, 2414, 2415, 2419, 2421, 2422
`\splitfirstmarks`
 968, 1040, 2390, 2391, 2398, 2399
`\splittopskip` 963, 1035
`\stanza@count` 1580, 1594, 1606
`\stanza@hang` 1582, 1614
`\stanza@modulo` 1580, 1609
`\stanzaindentbase`
 1560, 1570, 1607, 1610
`\startlock` 651
`\startstanzahook` 1578

- `\startsub` 618
`\sub@action` 308, 457, 557
`\sub@change` 61, 302, 303, 309
`\sub@lock` 1090
`\sub@lockR` 58, 317, 319, 321,
 324, 475, 481, 482, 484, 485,
 515, 516, 518, 1072, 1128, 1130,
 1131, 1133, 1189, 1229, 1231, 1233
`\sub@off` 626, 627
`\sub@on` 620, 621
`\subline@num` 235, 358, 376,
 377, 379, 409, 449, 1091, 1096, 1443
`\subline@numR` 236,
 240, 325, 329, 348, 370, 371,
 373, 402, 440, 563, 567, 1073,
 1078, 1149, 1156, 1243, 1244, 1439
`\sublinenumincrement` 5, 197
`\sublinenumincrement*` 5, 197
`\sublinenumr@p` . 1386, 1390, 1439, 1443
`\sublinenumrepR` 223, 236
`\sublines@false` 59, 306, 1118
`\sublines@true` 304, 1116
`\sublock@disp` 1191, 1195, 1200
`\symlinenum` 1385
`\sza@penalty` 1589, 1593
- T**
- `\temp` 1905, 1906, 1909–1912,
 1925, 1926, 1931–1933, 1942,
 1944, 1947–1950, 1954, 1955,
 1958–1961, 1969, 1971, 1974–1977
`\temp@` 1004, 1005
`\temp@spacing` 2431, 2432
`\tempa` 1943, 1945–1947, 1970, 1972–1974
`\textwidth` 18, 20, 753, 754
`\theledlanguageL` ... 1688, 1784, 2033
`\theledlanguageR` ... 1688, 1795, 2064
`\thepage` 615, 617, 1409, 1421
`\thepstart` 771, 791
`\thepstartL`
 . 8, 771, 811, 848, 855, 1329, 1334
`\thepstartR`
 . 8, 791, 814, 891, 898, 1341, 1346
`\thr@@` .. 484, 493, 516, 523, 1123, 1131
`\topskip` 2121
- U**
- `\unhbox` 1624,
 1807, 1812, 2045, 2048, 2076, 2079
- `\unhnamebox` 1619
`\unvbox` 969, 1041, 1626
`\unvnamebox` 1619
`\usebox` 1809, 1814
`\usernamecount` 1581, 1588, 1630, 1662,
 2012, 2281, 2284, 2308, 2310,
 2330, 2333, 2353, 2355, 2389, 2397
- V**
- `\value` 832, 876, 1605,
 1763, 1764, 1986, 1987, 2294, 2343
`\vbadness` 962, 1034
`\vbfnoteX` 1525, 1536
`\vbox` 843, 887, 1792, 1803
`\vl@dbfnote` 1505, 1509
`\vsplit` 966, 1038
- W**
- `\wd` 996
`\widthliketwocolumns` 4
`\widthliketwocolumnstrue` 11
`\WithSuffix` 172, 217–220, 1447
`\writtenlinesLfalse` 2003, 2292
`\writtenlinesLtrue` 2289
`\writtenlinesRfalse` 2004, 2341
`\writtenlinesRtrue` 2338
- X**
- `\x@lemma` 681–683, 704–706
`\xifinlist` 979,
 1005, 1051, 1786, 1797, 2040, 2071
`\xifinlistcs` 1873–
 1876, 1879–1882, 2222, 2223,
 2227, 2228, 2234, 2235, 2239, 2240
`\Xnoteswidthliketwocolumns` 4
`\xpg@main@language` 1720, 1721
`\xright@appenditem`
 430, 431, 433, 434,
 438, 445, 447, 454, 459, 461,
 463, 466, 468, 470, 478, 480,
 489, 512, 514, 521, 540, 541,
 551, 565, 577, 581, 585, 588,
 1438, 1442, 1505, 1509, 1525, 1536
- Z**
- `\zz@@@` 1401, 1413

Change History

v0.1		
General: First public release	1	
v0.10		
General: <code>\edlabel</code> commands on		
the right side are now correctly		
indicated.	1	
<code>\edlabel</code> commands which start		
a paragraph are now put in the		
right place.	1	
v0.11		
General: Change <code>\do@lineL</code> and		
<code>\do@lineR</code> to allow line num-		
bering by <code>pstart</code> (like in <code>eledmac</code>		
0.15).	41	
Lineation can be by <code>pstart</code> (like		
in <code>eledmac</code> 0.15).	17	
New management of hang-		
ingsymbol insertion, preventing		
undesirable insertions.	56	
Prevent shift of column separator		
when a verse is hanged	56	
<code>\affixline@numR</code> : Changed		
<code>\affixline@numR</code> to allow to		
disable line numbering (like in		
<code>eledmac</code> 0.15).	46	
<code>\Columns</code> : Line numbering by		
<code>pstart</code>	64	
<code>\get@nextboxR</code> : Change <code>\get@nextboxL</code>		
and <code>\get@nextboxR</code> to allow to		
disable line numbering (like in		
<code>eledmac</code> 0.15).	77	
<code>Pstart</code> number can be printed in		
side	77	
v0.12		
General: New new management of		
hangingsymbol insertion, pre-		
venting undesirable insertions. .	56	
v0.2		
General: Added section of babel re-		
lated code	60	
Fix babel problems	1	
<code>\Columns</code> : Added <code>\l@duselang</code>		
and <code>\l@duselanguage</code> to		
<code>\Columns</code>	64	
<code>\Pages</code> : Added <code>\l@duselanguage</code>		
to <code>\Pages</code>	70	
v0.3		
General: Added <code>\do@linehook</code>		
and <code>\do@lineRhook</code>	42	
Reorganize for <code>ledarab</code>	1	
<code>\affixline@numR</code> : Changed		
<code>\affixline@numR</code> to match new		
<code>eledmac</code>	46	
<code>\do@actions@nextR</code> : Used		
<code>\do@actions@fixedcode</code> in		
<code>\do@actionsR</code>	44	
<code>\do@lineL</code> : Added <code>\do@linehook</code>		
to <code>\do@lineL</code>	41	
Simplified <code>\do@lineL</code> by using		
macros for some common code .	41	
<code>\do@lineR</code> : Changed <code>\do@lineR</code>		
similarly to <code>\do@lineL</code>	42	
<code>Leftside</code> : Added hooks into Left-		
side environment	35	
<code>\flag@end</code> : Removed extraneous		
spaces from <code>\flag@end</code>	31	
<code>\ifledRcol</code> : Moved <code>\ifl@dpairing</code>		
to <code>eledmac</code>	13	
<code>\ifpst@rtedR</code> : Moved <code>\ifpst@rtedL</code>		
to <code>eledmac</code>	14	
<code>\l@ddlinenumR</code> : Simplified		
<code>\leftlinenumR</code> and <code>\rightlinenumR</code>		
by introducing <code>\l@ddlinenumR</code> .	20	
<code>\l@dnumpstartsR</code> : Moved		
<code>\l@dnumpstartsL</code> to <code>eledmac</code> .	59	
<code>\ledsavedprintlines</code> : Simpli-		
fied <code>\printlinesR</code> by using		
<code>\setprintlines</code>	52	
<code>\ledstrutR</code> : Added <code>\ledtrutL</code> and		
<code>\ledstrutR</code>	72	
<code>\normalbfnoteX</code> : Removed		
extraneous spaces from		
<code>\normalbfnoteX</code>	55	
<code>\Pages</code> : Added <code>\ledstrutL</code> to		
<code>\Pages</code>	70	
Added <code>\ledstrutR</code> to <code>\Pages</code> .	71	

	<code>\Rightsidehookend:</code>	Added	Possibility to number the	
	<code>\Leftsidehook, \Leftsidehookend,</code>		<code>pstart</code> with the commands	
	<code>\Rightsidehook and \Rightsidehookend</code>		<code>\numberpstarttrue</code>	1
	35	<code>\ifledRcol:</code> Moved <code>\iflledRcol</code>	
	<code>\sublinenumrepR:</code>	Added	and <code>\ifnumberingR</code> to <code>eledmac</code>	13
	<code>\linenumrepR and \sublinenumrepR</code>	v0.9.1		
	19	General: The numbering of the	
v0.3.a	General: Minor <code>\linenummargin</code>		<code>pstarts</code> restarts on each	
	fix	1	<code>\beginnumbering</code>	1
v0.3.b	General: Improved parallel page			
	balancing	1	General: Debug : with <code>\Columns</code> ,	
v0.3.c	General: Compatibility with Poly-		the hanging indentation now	
	glossia	1	runs on the left columns and the	
v0.3.a	General: Compatibility with Poly-		hanging symbol is shown only	
	glossia	1	when <code>\stanza</code> is used.	1
v0.3a	<code>\line@marginR:</code> Don't just		v0.9.3	
	set <code>\line@marginR</code> in		General: <code>\thepstartL</code> and	
	<code>\linenummargin</code>	18	<code>\thepstartR</code> use now	
v0.3b	<code>\Pages:</code> Added <code>\l@dminpagelines</code>		<code>\bfseries</code> and not <code>\bf</code> , which	
	calculation for succeeding page		is deprecated and makes con-	
	pairs	72	flicts with memoir class.	1
v0.4	General: No more <code>ledparpatch</code> . All		v1.0	
	patches are now in the main		General: Compatibility with <code>eled-</code>	
	file.	1	<code>mac</code> . Change name to <code>eledpar</code> . .	1
v0.5	General: Corrections about		Debug in lineation by <code>pstart</code> ..	17
	<code>\section</code> and other titles in		v1.0.1	
	numbered sections	1	General: Correction on <code>\numberonlyfirstinline</code>	
v0.6	General: Be able to use <code>\chapter</code> in		with lineation by <code>pstart</code> or by	
	parallel pages.	1	page.	1
v0.7	General: Option 'shiftedverses'		v1.1	
	which make there is no blank		General: <code>Shiftedverses</code> becomes	
	between two parallel verses with		<code>shiftedpstarts</code>	1
	inequal length.	1	<code>\pstartR:</code> Add <code>\labelpstarttrue</code>	
v0.8	General: Possibility to have a sym-		(from <code>eledmac</code>).	37
	bol on each hanging of verses,		v1.1.1	
	like in the french typogra-		<code>\pstartR:</code> Correct <code>\pstartR</code> bug in-	
	phy. Redefine the commande		troduced by 1.1.	37
	<code>\hangingsymbol</code> to define the		v1.1.2	
	character.	1	<code>\affixside@noteR:</code> Remove spuri-	
v0.9	General: Possibility to number		ous space between line number	
	<code>\pstart</code>	8	and line content	54
			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
			v1.3	
			General: Manage RTL language. .	32

v1.3.2	General: Debug with some classes.	1	change both Left and Right-side.	19
v1.3.3	General: Debugging the left notes of the right column.	54	v1.6.0	General: Add tool and documentation for parallel ledgroups . . .
	<code>\l@dbfnote</code> : Spurious space with footnote in right column.	55	v1.7.0	General: Add, as in <code>eledmac</code> , features to make crossrefs with <code>pstart</code> numbers.
v1.3.4	General: Allow use of commands in sidenotes, as introduced by <code>eledmac</code> 1.0.	54	v1.8.0	General: <code>\beginnumbering</code> is defined only on <code>eledmac</code> , not on <code>eledpar</code>
v1.3.5	<code>\normalbfnoteX</code> : Allows one to redefine <code>\thefootnoteX</code> with <code>alph</code> when some packages are loaded.	55		<code>\l@dlsnote</code> , <code>\l@drsnote</code> and <code>\l@dcsnote</code> defined only one time, in <code>eledmac</code>
v1.4	General: Added <code>\do@insidelineLhook</code> and <code>\do@insidelineRhook</code> . . .	42		Add <code>\beforecolumnseparator</code> and <code>\aftercolumnseparator</code> . . .
v1.4.1	<code>\normalbfnoteX</code> : Fix bug with normal familiar footnotes when mixing RTL and LTR text. . .	55		Add <code>\columnspan</code>
	<code>astanza</code> : Enable the use of <code>stanza</code> environment.	57		Add, as in <code>eledmac</code> , new system of sectioning commands.
v1.4.2	<code>\line@list@stuffR</code> : Open / close immediately the line-list file when in <code>minipage</code> , except if the <code>minipage</code> is a <code>ledgroup</code>	30		Add, as in <code>eledmac</code> , option to insert something after <code>\pends</code> / verses.
v1.4.3	General: Corrects a false hanging verse when a verse is exactly the length of a line.	1		Add, as in <code>eledmac</code> , option to insert something between <code>\pstarts</code> / verse.
	<code>\inserthangingsymbolR</code> : Hang verse is now not automatically flush right.	56		Change <code>\do@lineR</code> and <code>\do@lineL</code> to allow new sectioning commands.
	<code>\pendL</code> : Spurious spaces in <code>\pendL</code>	39		Compatibility with <code>musixtex</code> . . .
	<code>\pendR</code> : Spurious spaces in <code>\pstartR</code>	40		Debug <code>eledmac</code> sectioning command after using <code>\resumenummering</code>
	<code>\pstartR</code> : Spurious spaces in <code>\pstartL</code> and <code>\pstartR</code>	37		New sectioning commands, as in <code>eledmac</code>
v1.5.0	General: Add, as in <code>eledmac</code> , features to manage page breaks. . .	1		<code>\Columns</code> : Modify <code>\Columns</code> to enable to add section's title. . . .
	<code>\sublinenumincrement*</code> : Add starred version of <code>\firstlinenum</code> , <code>\linenumincrement</code> , <code>\firstsublinenum</code> , <code>\Pages</code> : Modify <code>\Pages</code> to enable to add section's title.	69		Suppress <code>\l@dchecklang</code> from <code>\Columns</code>
				<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.

<code>\pendL</code> : As in <code>eledmac</code> , <code>\pendL</code> can have an optional argument. . .	39	<code>eledpar</code> send to <code>eledpar</code> handbook	14
<code>\pendR</code> : As in <code>eledmac</code> , <code>\pendR</code> can have an optional argument. . .	40	<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
<code>\print@columnseparator</code> : Move some code of <code>\Columns</code> to <code>\print@columnseparator</code> . . .	65	<code>\lineation*</code> : Add <code>\lineation*</code> .	18
<code>\pstartR</code> : As in <code>eledmac</code> , <code>\pendL</code> and <code>\pendR</code> can have an optional argument.	37	v1.8.3	
<code>\sidenotemargin*</code> : <code>\sidenotemargin</code> is now directly defined in <code>eledmac</code> to be able to manage <code>eledpar</code>	54	General: Add <code>\noeledxxx</code> , as in <code>eledmac</code>	1
Add <code>\sidenotemargin*</code>	54	<code>\doinsidelineRhook</code> : Added <code>\dolineLhook</code> , <code>\dolineRhook</code> , <code>\doinsidelineLhook</code> and <code>\doinsidelineRhook</code>	42
<code>\theledlanguageR</code> : Correct left/right language setting with <code>polyglossia</code>	62	<code>\Pages</code> : Debug blank pages when using optional argument in the last <code>\pend</code>	69
v1.8.1		<code>\resumenumberingR</code> : Debug <code>\resumenumberingR</code>	16
<code>\do@lineL</code> : Fix a bug with critical notes a the begining of a page, (maybe added by v1.8.0) (?). .	41	v1.9.0	
<code>\do@lineR</code> : Fix a bug with critical notes a the begining of a page, added by v1.8.0 (?).	42	General: Add <code>\AtBeginPairs</code> macro.	4
v1.8.2		Compatibility with <code>\Xnoteswidthliketwocolumns</code> and <code>\notesXwidthliketwocolumns</code>	1
General: Debug <code>\eledxxx</code> with some paper sizes	1	<code>\ifwidthliketwocolumns</code> : Added <code>widthliketwocolumns</code> option . .	13
Debug left and side note (bugs added by 1.8.0)	1	<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.	62
<code>\eledpar@error</code> : Errors specific to			