

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 v1.10.1, last revised 2015/01/16.

[†]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	15
13 Sectioning commands	16
14 Line counting	19
14.1 Choosing the system of lineation	19
14.2 Line-number counters and lists	22
14.3 Reading the line-list file	23
14.4 Commands within the line-list file	24
14.5 Writing to the line-list file	32
15 Marking text for notes	34
16 Parallel environments	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	54
18.1 Normal footnote formatting	54
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	82
29 Page break/no page break, depending on the specific line	82
30 Parallel ledgroup	83
31 The End	86
Appendix A Some things to do when changing version	87
Appendix A.1 Migration to eledpar 1.4.3	87
References	87
Index	87
Change History	98

List of Figures

1 Introduction

The EDMAC macros [LW90] for typesetting critical editions of texts have been available for use with TeX for some years. Since EDMAC became available there had been a small but constant demand for a version of EDMAC that could be used with L^AT_EX. The eledmac package was introduced in 2003 in an attempt to satisfy that request.

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

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

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

2 The `eledpar` package

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

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

2.1 General

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

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

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

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

```
\maxchunks{5120}
```

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

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

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

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

3 Parallel columns

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

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

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

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

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

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

	There is no required pagebreak before or after the columns.
<code>\Lcolwidth</code>	The lengths <code>\Lcolwidth</code> and <code>\Rcolwidth</code> are the widths of the left and right
<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>\setstanzaindent</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>
<code>\widthliketwocolumns</code>	If you want to come back to the previous behavior, just set them with
	a negative value. If you want to mix texts in columns and text 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 columns. <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}
\begin{Leftside} ... \end{Leftside}
...
\end{pages}
\Pages
```

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

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

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

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

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

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

5 Left and right texts

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

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

Within these environments you can designate the line numbering scheme(s) to be used. The `eledmac` package originally used counters for specifying the numbering scheme; now both `eledmac`¹ and the `eledpar` package use macros instead. Following `\firstlinenum{<num>}` the first line number will be `<num>`, and following `\linenumincrement{<num>}` only every `<num>`th line will have a printed number. Using these macros inside the `Leftside` and `Rightside` environments

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

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

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

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

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

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

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

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

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

6 Numbering text lines and paragraphs

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

```
\beginnumbering
<text>
```

```
\endnumbering
```

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

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

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

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

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

Its default definition is:

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

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

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

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

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

7 Verse

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

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

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

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

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

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

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

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

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

  \end{astanza}
  ...
```

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

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

    \strut &
```

```

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

```

`\hangingsymbol` Like in `eledmac`, you could redefine the command `\hangingsymbol` to insert a character in each hanging line. If you use it, you must run \LaTeX two time. Example for the French typography

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

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

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

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

8 Side notes

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

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

9 Parallel ledgroups

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

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

See the following example:

```

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

```

```

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

```

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

```

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

```

9.1 Parallel ledgroups and `setspace` package

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

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

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

10 Sectioning commands

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

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

11 Implementation overview

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

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

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

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

12 Preliminaries

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

```

1 (*code)
2 \NeedsTeXFormat{LaTeX2e}
3 \ProvidesPackage{eledpar}[2015/01/16 v1.10.1 eledmac extension for parallel texts]%
4

```

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

`\ifshiftedpstarts`

```

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

```

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

```

11 \DeclareOption{widthliketwocolumns}{\widthliketwocolumnstrue}%

```

```

12 \ProcessOptions%

```

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

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

```

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

```

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

```

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

```

12.1 Messages

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

`\eledpar@error`

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

`\led@err@BadLeftRightPstarts`

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

`\led@err@LeftOnRightPage`

`\led@err@RightOnLeftPage`

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

13 Sectioning commands

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

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

```
36 \newcount\section@numR
37 \section@numR=\z@
```

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

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

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

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



```

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

```

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

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

```

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

```

```

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

```

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

```

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

```

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

```

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

```

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

```

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

```

```

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

```

14 Line counting

14.1 Choosing the system of lineation

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

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

`\bypstart@Rtrue`

`\bypstart@Rfalse`

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

```

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

```

`\lineationR` `\lineationR{<word>}` is the macro used to select the lineation system for right texts. Its argument is a string: either `page`, `pstart` or `section`.

```

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

```

```

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

```

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

```

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

```

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

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

```

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

```

By default put right text numbers at the right.

```

185 \line@marginR=\@ne
186

```

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

```

187 \newcounter{firstlinenumR}

```

```

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

```

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

```

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

```

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

```

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

```

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

```

220 \newcommand*\Rlineflag{R}
221

```

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

```

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

```

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

```

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

14.2 Line-number counters and lists

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

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

```

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

`\line@listR` `\insertlines@listR` `\actionlines@listR` `\actions@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.

Here are the commands to create these lists:

```

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

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

```

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

```

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

```

254 \newcount\page@numR
255

```

14.3 Reading the line-list file

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

```

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

```

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

```

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

```

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

```

271   \list@clear{\maxlinesinpar@list}

```

Now get the file and interpret it.

```

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

```

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

```

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

```

```

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

```

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

14.4 Commands within the line-list file

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

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

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

```

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

```



```

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

```

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

```

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

```

```

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

```

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

```

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

```

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

```

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

```

```

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

```

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

```

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

```

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

```

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

```

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

```

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

```

```

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

```

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

```

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

```

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

```

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

```

```

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

```

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

```

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

```

```

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

```

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

```

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

```

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

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

```

549 \newcount\insert@countR

```

- #2, a sequence of other line-list-file commands, executed to determine the ending line-number. (This may also include other `\@ref` commands, corresponding to uses of `\edtext` within the first argument of another instance of `\edtext`.)

The first thing `\@ref` itself does is to add the specified number of items to the `\insertlines@list` list.

```

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

```

Next, process the second argument to determine the page and line numbers for the end of this lemma. We temporarily equate `\@ref` to a different macro that just executes its argument, so that nested `\@ref` commands are just skipped this time. Some other macros need to be temporarily redefined to suppress their action.

```

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

```

```

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

```

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

```

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

```

Finally, execute the second argument of `\@ref` again, to perform for real all the commands within it.

```

575 #2
576 \else

```

And when not in right text

```

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

```

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

```

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

```

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

```

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

```

```

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

```

14.5 Writing to the line-list file

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

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

```

595 \newwrite\linenum@outR

```

`\iffirst@linenum@out@R` Once any file is opened on this stream, we keep it open forever, or else switch to another file that we keep open.

```

\first@linenum@out@Rtrue
\first@linenum@out@Rfalse
596 \newif\iffirst@linenum@out@R
597   \first@linenum@out@Rtrue

```

`\line@list@stuffR` This is the right text version of the `\line@list@stuff{<file>}` macro. It is called by `\beginnumberingR` and performs all the line-list operations needed at the start of a section. Its argument is the name of the line-list file.

```

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

```

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

```

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

```

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

```

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

```

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

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

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

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

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

`\setline` You can use `\setline{<num>}` in running text (i.e., within `\pstart... \pend`) to set the current visible line-number to a specified positive value.

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

`\setlinenum` You can use `\setlinenum{<num>}` before a `\pstart` to set the visible line-number to a specified positive value. It writes a `\l@d@set` command to the line-list file.

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

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

```

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

```

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

```

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

```

15 Marking text for notes

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

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

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

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

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

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

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

```

665 \long\def\critext#1#2/{\leavevmode
666   \@edtext@true%
667   \begingroup
668     \renewcommand{\@tag}{\no@expands #1}%
669     \set@line
670     \ifledRcol \global\insert@countR \z@
671     \else      \global\insert@count \z@ \fi

```

```

672 \ignorespaces #2\relax
673 \@ifundefined{xpg@main@language}{%if not polyglossia
674 \flag@start}%
675 {\if@RTL\flag@end\else\flag@start\fi% be careful on the direction of writing with polyglossia
676 }%
677 \endgroup
678 \showlemma{#1}%
679 \ifx\end@lemmas\empty \else
680 \gl@p\end@lemmas\to\x@lemma
681 \x@lemma
682 \global\let\x@lemma=\relax
683 \fi
684 \@ifundefined{xpg@main@language}{%if not polyglossia
685 \flag@end}%
686 {\if@RTL\flag@start\else\flag@end\fi% be careful on the direction of writing with polyglossia
687 }
688 \@edtext@false%
689 \@lemmacommand@false%
690 }

```

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

```

691 \renewcommand{\edtext}[2]{\leavevmode
692 \@edtext@true%
693 \begingroup%
694 \renewcommand{\@tag}{\no@expands #1}%
695 \set@line%
696 \ifledRcol \global\insert@countR \z@%
697 \else \global\insert@count \z@ \fi%
698 \ignorespaces #2\relax%
699 \@ifundefined{xpg@main@language}{%if not polyglossia
700 \flag@start}%
701 {\if@RTL\flag@end\else\flag@start\fi% be careful on the direction of writing with polyglossia
702 }%
703 \endgroup%
704 \showlemma{#1}%
705 \ifx\end@lemmas\empty \else%
706 \gl@p\end@lemmas\to\x@lemma%
707 \x@lemma%
708 \global\let\x@lemma=\relax%
709 \fi%
710 \@ifundefined{xpg@main@language}{%if not polyglossia
711 \flag@end}%
712 {\if@RTL\flag@start\else\flag@end\fi% be careful on the direction of writing with polyglossia
713 }%
714 \@edtext@false%
715 \@lemmacommand@false%
716 }
717

```

`\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`.

```

718 \renewcommand*{\set@line}{%
719   \ifledRcol
720     \ifx\line@listR\empty
721       \global\noteschanged@true
722       \xdef\l@d@nums{000|000|000|000|000|000|\edfont@info}%
723     \else
724       \gl@p\line@listR\to\@tempb
725       \xdef\l@d@nums{\@tempb|\edfont@info}%
726       \global\let\@tempb=\undefined
727     \fi
728   \else
729     \ifx\line@list\empty
730       \global\noteschanged@true
731       \xdef\l@d@nums{000|000|000|000|000|000|\edfont@info}%
732     \else
733       \gl@p\line@list\to\@tempb
734       \xdef\l@d@nums{\@tempb|\edfont@info}%
735       \global\let\@tempb=\undefined
736     \fi
737   \fi}
738
```

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
pages   parallel pages.
chapterinpages 739 \newenvironment{pairs}{%}
740   \l@dpairingtrue
741   \l@dpagingfalse
742   \initnumbering@sectcmd
743   \at@begin@pairs%
744 }{%
745   \l@dpairingfalse
746 }
747

\AtBeginPairs The \AtBeginPairs macro just define a \at@begin@pairs macro, called at the
begining of each pairs environments.
748 \newcommand{\AtBeginPairs}[1]{\xdef\at@begin@pairs{#1}}%
749 \def\at@begin@pairs{}%
750
```

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.

```

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

```

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

```

769 \newif\ifinstanzaL
770 \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.

```

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

```

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

```

\Rightsidehook 783 \newcommand*{\Leftsidehook}{}
\Rightsidehookend 784 \newcommand*{\Leftsidehookend}{}
785 \newcommand*{\Rightsidehook}{}
786 \newcommand*{\Rightsidehookend}{}

```

787

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.

```

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

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.

```

809 \newcount\num@linesR
810 \newbox\one@lineR
811 \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.

```

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

```

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

```

835 \ifpstart@rtedL\else%
836   \setcounter{pstartLold}{\value{pstartL}}%
837   \list@clear{\inserts@list}%
838   \global\let\next@insert=\empty%
839   \global\pstart@rtedLtrue%
840 \fi%
841 \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.

```

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

Dump the optional arguments
860 \ifstrempy{#1}%
861 {\csgdef{before@pstartL@the\l@dnumstartsL}{\at@every@pstart}}%
862 {\csgdef{before@pstartL@the\l@dnumstartsL}{\noindent#1}}%
863 }

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

```



```

887 \ifnum\l@dnumpstartsR>\l@dc@maxchunks%
888   \led@err@TooManyPstarts%
889   \global\l@dnumpstartsR=\l@dc@maxchunks%
890 \fi%
891 \global\setnamebox{l@dRcolrawbox\the\l@dnumpstartsR}=\vbox\bgroup%
892   \ifautopar\else%
893     \ifnumberpstart%
894       \ifsidepstartnum\else%
895         \thepstartR%
896       \fi%
897     \fi%
898   \fi%
899 \hsize=\Rcolwidth%
900 \numberedpar@true%
901 \iflabelpstart\protected@edef\@currentlabel%
902   {\p@pstartR\thepstartR}\fi%
903 \ifstrempy{#1}%
904   {\csgdef{before@pstartR@the\l@dnumpstartsR}{\at@every@pstart}}%
905   {\csgdef{before@pstartR@the\l@dnumpstartsR}{\noindent#1}}%
906 }

```

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

```

907 \newcommandx*\pendL}[1][1]{%
908   \ifnumbering \else%
909     \led@err@PendNotNumbered%
910   \fi%
911   \ifnumberedpar@ \else%
912     \led@err@PendNoPstart%
913   \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.

```

914 \l@dzeropenalties%
915 \endgraf\global\num@lines=\prevgraf\egroup%
916 \global\par@line=0%

```

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

```

917 \endgroup%
918 \ignorespaces%
919 \@oldnobreak%
920 \ifnumberpstart%
921   \addtocounter{pstartL}{1}%
922 \fi
923 \parledgroup@beforenotes@save{L}%

```

Dump content of the optional argument.

```

924 \ifstrempy{#1}%

```

```

925     {\csgdef{after@pendL@the\l@dnumpstartsL}{\at@every@pend}}%
926     {\csgdef{after@pendL@the\l@dnumpstartsL}{\noindent#1}}%
927  }

```

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

```

928 \newcommandx*{\pendR}[1][1]{%
929   \ifnumberingR \else%
930     \led@err@PendNotNumbered%
931   \fi%
932   \ifnumberedpar@ \else%
933     \led@err@PendNoPstart%
934   \fi%
935   \l@dzero penalties%
936   \endgraf\global\num@linesR=\prevgraf\egroup%
937   \global\par@lineR=0%
938   \endgroup%
939   \ignorespaces%
940   \@oldnobreak%
941   \ifnumberpstart%
942     \addtocounter{pstartR}{1}%
943   \fi%
944   \parledgroup@before notes@save{R}%
945   \ifstrepty{#1}%
946     {\csgdef{after@pendR@the\l@dnumpstartsR}{\at@every@pend}}%
947     {\csgdef{after@pendR@the\l@dnumpstartsR}{\noindent#1}}%
948 }
949

```

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

```

950 \newif\ifprint@last@after@pendL%
951 \newif\ifprint@last@after@pendR%

```

17.2 Processing one line

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

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

```

952 \newbox\l@dleftbox
953 \newbox\l@drightbox
954

```

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

```

\countRline 955 \newcount\countLline
956   \countLline \z@
957 \newcount\countRline
958   \countRline \z@

```

959

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

`\@donetotallinesR` 960 `\newcount\@donereallinesL`
 961 `\newcount\@donetotallinesL`
 962 `\newcount\@donereallinesR`
 963 `\newcount\@donetotallinesR`
 964

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

```

965 \newcommand*{\do@lineL}{%
966   \advance\countLline \@ne
967   \ifvbox\namebox{1@dLcolrawbox\the\l@dpscL}%
968   {\vbadness=10000
969     \splittopskip=\z@
970     \do@lineLhook
971     \l@emptyd@ta
972     \global\setbox\one@line=\vsplit\namebox{1@dLcolrawbox\the\l@dpscL}
973       to\baselineskip}%
974   \IfStrEq{\splitfirstmarks\parledgroup@}{begin}{\parledgroup@notes@startL}{%
975     \unvbox\one@line \global\setbox\one@line=\lastbox
976     \getline@numL
977     \ifnum\@lock>\@ne%
978       \inserthangingsymboltrue%
979     \else%
980       \inserthangingsymbolfalse%
981     \fi
982     \setbox\l@dleftbox
983     \hb@xt@ \Lcolwidth{%
984       \affixline@num
985       \xifinlist{\the\l@dpscL}{\eled@sections@@}%
986       {\add@inserts\affixside@note}%
987       {\print@lineL}}%
988     \add@penaltiesL
989     \global\advance\@donereallinesL\@ne
990     \global\advance\@donetotallinesL\@ne
991   \else
992     \setbox\l@dleftbox \hb@xt@ \Lcolwidth{\hspace*{\Lcolwidth}}%
993     \global\advance\@donetotallinesL\@ne
994   \fi}
995
996
```

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

```

997 \def\print@lineL{%

```

```

998 \affixpstart@numL%
999 \l@dld@ta %space kept for backward compatibility
1000 \add@inserts\affixside@note%
1001 \l@dlsn@te %space kept for backward compatibility
1002 {\ledllfill\hb@xt@ \wd\one@line{\do@insidelineLhook\inserthangingsymbolL\new@lineL\l@
1003 \l@drsn@te}}
1004

```

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

```

1005 \def\print@eledsectionL{%%
1006 \addtocounter{pstartL}{-1}%
1007 \ifdefstring{\@eledsectnotoc}{L}{\ledsectnotoc}{%
1008 \ifdefstring{\@eledsectmark}{L}{\ledsectnomark}%
1009 \numdef{\temp@}{\l@dpscL-1}%
1010 \xifinlist{\temp@}{\eled@sections@@}{\@nobreaktrue}{\@nobreakfalse}%
1011 \@eled@sectioningtrue%
1012 \csuse{eled@sectioning@the\l@dpscL}%
1013 \@eled@sectioningfalse%
1014 \global\csundef{eled@sectioning@the\l@dpscL}%
1015 \if@RTL%
1016 \hspace{-3\paperwidth}%
1017 {\hbox{\l@dunhbox@line{\one@line}} \new@line}%
1018 \else%
1019 \hspace{3\paperwidth}%
1020 {\new@line \hbox{\l@dunhbox@line{\one@line}}}%
1021 \fi%
1022 \vskip\eledsection@correcting@skip%
1023 }
1024

```

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

```

\doininsidelineLhook 1025 \newcommand*{\dolineLhook}[1]{\gdef\dolineLhook{#1}}%
\doininsidelineRhook 1026 \newcommand*{\dolineRhook}[1]{\gdef\dolineRhook{#1}}%
1027 \newcommand*{\doininsidelineLhook}[1]{\gdef\doininsidelineLhook{#1}}%
1028 \newcommand*{\doininsidelineRhook}[1]{\gdef\doininsidelineRhook{#1}}%
1029

```

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

```

\dolineRhook 1030 \newcommand*{\do@lineLhook}{}
\doininsidelineLhook 1031 \newcommand*{\do@lineRhook}{}
\doininsidelineRhook 1032 \newcommand*{\do@insidelineLhook}{}
1033 \newcommand*{\do@insidelineRhook}{}
1034

```

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

```

1035 \newcommand*{\do@lineR}{%

```

```

1036 \ledRcol@true%
1037 \advance\countRline \@ne
1038 \ifvbox\namebox{l@dRcolrawbox\the\l@dpscR}%
1039 {\vbadness=10000
1040 \splittopskip=\z@
1041 \do@lineRhook
1042 \l@emptyd@ta
1043 \global\setbox\one@lineR=\vsplit\namebox{l@dRcolrawbox\the\l@dpscR}
1044 to\baselineskip}%
1045 \IfStrEq{\splitfirstmarks\parledgroup@}{\begin}{\parledgroup@notes@startR}{\}
1046 \unvbox\one@lineR \global\setbox\one@lineR=\lastbox
1047 \getline@numR
1048 \ifnum\@lockR>\@ne%
1049 \inserthangingsymbolRtrue
1050 \else%
1051 \inserthangingsymbolRfalse%
1052 \fi%
1053 \setbox\l@dtrightbox
1054 \hb@xt@ \Rcolwidth{%
1055 \affixline@numR%
1056 \xifinlist{\the\l@dpscR}{\eled@sectionsR@@}%
1057 {\add@insertsR\affixside@noteR}%
1058 {\print@lineR}%
1059 }%
1060 \add@penaltiesR
1061 \global\advance\@donereallinesR\@ne
1062 \global\advance\@donetotallinesR\@ne
1063 \else
1064 \setbox\l@dtrightbox \hb@xt@ \Rcolwidth{\hspace*\Rcolwidth}
1065 \global\advance\@donetotallinesR\@ne
1066 \fi
1067 \ledRcol@false%
1068 }
1069
1070

```

```

\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.

```

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

```

```

1078     \global\advance\subline@numR \@ne
1079     \fi
1080   \else
1081     \ifnum\@lockR<\tw@
1082       \global\advance\line@numR \@ne
1083       \global\subline@numR \z@
1084     \fi
1085   \fi
1086 \fi
1087 \fi
1088 }
1089 \newcommand*{\getline@numL}{%
1090   \global\advance\absline@num \@ne
1091   \do@actions
1092   \do@ballast
1093 \ifledgroupnotesL@\else\ifnumberline
1094   \ifsublines@
1095     \ifnum\sub@lock<\tw@
1096       \global\advance\subline@num \@ne
1097     \fi
1098   \else
1099     \ifnum\@lock<\tw@
1100       \global\advance\line@num \@ne
1101       \global\subline@num \z@
1102     \fi
1103   \fi
1104 \fi
1105 \fi
1106 }
1107
1108

```

`\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.

```

1109 \newcommand*{\do@ballastR}{\global\ballast@count=\z@
1110   \begingroup
1111     \advance\absline@numR \@ne
1112     \ifnum\next@actionlineR=\absline@numR
1113       \ifnum\next@actionR>-1001
1114         \global\advance\ballast@count by -\c@ballast
1115       \fi
1116     \fi
1117   \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.

```

1118 \newcommand*{\do@actions@fixedcodeR}{%
1119   \ifcase\@l@dttempcnta%
1120     \or%                               % 1001
1121     \global\sublines@true
1122   \or%                               % 1002
1123     \global\sublines@false
1124   \or%                               % 1003
1125     \global\@lockR=\@ne
1126   \or%                               % 1004
1127     \ifnum\@lockR=\tw@
1128       \global\@lockR=\thr@@
1129     \else
1130       \global\@lockR=\z@
1131     \fi
1132   \or%                               % 1005
1133     \global\sub@lockR=\@ne
1134   \or%                               % 1006
1135     \ifnum\sub@lockR=\tw@
1136       \global\sub@lockR=\thr@@
1137     \else
1138       \global\sub@lockR=\z@
1139     \fi
1140   \or%                               % 1007
1141     \l@dskipnumbertrue
1142   \else
1143     \led@warn@BadAction
1144   \fi}
1145
1146
1147 \newcommand*{\do@actionsR}{%
1148   \global\let\do@actions@nextR=\relax
1149   \@l@dttempcntb=\absline@numR
1150   \ifnum\@l@dttempcntb<\next@actionlineR\else
1151     \ifnum\next@actionR>-1001\relax
1152       \global\page@numR=\next@actionR
1153       \ifbypage@R
1154         \global\line@numR \z@ \global\subline@numR \z@
1155       \fi
1156     \else
1157       \ifnum\next@actionR<-4999\relax % 9/05 added relax here
1158         \@l@dttempcnta=-\next@actionR
1159         \advance\@l@dttempcnta by -5001\relax
1160         \ifsublines@
1161           \global\subline@numR=\@l@dttempcnta
1162         \else
1163           \global\line@numR=\@l@dttempcnta
1164         \fi
1165       \else
1166         \@l@dttempcnta=-\next@actionR
1167         \advance\@l@dttempcnta by -1000\relax

```

```

1168     \do@actions@fixedcodeR
1169     \fi
1170 \fi
1171 \ifx\actionlines@listR\empty
1172     \gdef\next@actionlineR{1000000}%
1173 \else
1174     \gl@p\actionlines@listR\to\next@actionlineR
1175     \gl@p\actions@listR\to\next@actionR
1176     \global\let\do@actions@nextR=\do@actionsR
1177 \fi
1178 \fi
1179 \do@actions@nextR}
1180

```

17.4 Line number printing

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

```

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

```



```

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

```

```

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

```

```

1301 \fi
1302 \ifodd\@l@tempcntb
1303 \gdef\l@drrd@ta{\rlap{\rightpstartnumL}}}%
1304 \else
1305 \gdef\l@dld@ta{\llap{\leftpstartnumL}}}%
1306 \fi
1307 \fi
1308 \fi
1309 }
1310 \newcommand*\affixpstart@numR}{%
1311 \ifsidepstartnum
1312 \if@twocolumn
1313 \if@firstcolumn
1314 \gdef\l@dld@ta{\llap{\leftpstartnumR}}}%
1315 \else
1316 \gdef\l@drrd@ta{\rlap{\rightpstartnumR}}}%
1317 \fi
1318 \else
1319 \l@tempcntb=\line@marginR
1320 \ifnum\@l@tempcntb>\@ne
1321 \advance\@l@tempcntb \page@numR
1322 \fi
1323 \ifodd\@l@tempcntb
1324 \gdef\l@drrd@ta{\rlap{\rightpstartnumR}}}%
1325 \else
1326 \gdef\l@dld@ta{\llap{\leftpstartnumR}}}%
1327 \fi
1328 \fi
1329 \fi
1330 }
1331
1332 \newcommand*\leftpstartnumL){
1333 \ifpstartnum
1334 \thepstartL
1335 \kern\linenumsep\global\pstartnumfalse\fi
1336 }
1337 \newcommand*\rightpstartnumL){
1338 \ifpstartnum\kern\linenumsep
1339 \thepstartL
1340 \global\pstartnumfalse\fi
1341 }
1342 \newif\ifpstartnumR
1343 \pstartnumRtrue
1344 \newcommand*\leftpstartnumR){
1345 \ifpstartnumR
1346 \thepstartR
1347 \kern\linenumsep\global\pstartnumRfalse\fi
1348 }
1349 \newcommand*\rightpstartnumR){
1350 \ifpstartnumR\kern\linenumsep

```

```

1351 \thepstartR
1352 \global\pstartnumRfalse\fi
1353 }

```

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.

```

1354 \list@create{\inserts@listR}

\add@insertsR The right text version.
\add@inserts@nextR
1355 \newcommand*{\add@insertsR}{%
1356   \global\let\add@inserts@nextR=\relax
1357   \ifx\inserts@listR\empty \else
1358     \ifx\next@insertR\empty
1359       \ifx\insertlines@listR\empty
1360         \global\noteschanged@true
1361         \gdef\next@insertR{100000}%
1362       \else
1363         \glp\insertlines@listR\to\next@insertR
1364       \fi
1365     \fi
1366     \ifnum\next@insertR=\absline@numR
1367       \glp\inserts@listR\to\@insertR
1368       \@insertR
1369       \global\let\@insertR=\undefined
1370       \global\let\next@insertR=\empty
1371       \global\let\add@inserts@nextR=\add@insertsR
1372     \fi
1373   \fi
1374 \add@inserts@nextR}
1375

```

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.

```

1376 \newcommand*{\add@penaltiesL}{ }
1377 \newcommand*{\add@penaltiesR}{ }
1378

```

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.

```

1379 \newcommand*{\flush@notesR}{%
1380   \@xloop
1381   \ifx\inserts@listR\empty \else
1382     \gl@p\inserts@listR\to\@insertR
1383     \@insertR
1384     \global\let\@insertR=\undefined
1385   \repeat}
1386

```

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
1387 \def\printlinesR#1|#2|#3|#4|#5|#6|#7|{\begingroup
1388   \setprintlines{#1}{#2}{#3}{#4}{#5}{#6}%
1389   \ifl@d@pnum #1\fullstop\fi
1390   \ifledplinum \linenumr@p{#2}\Rlineflag\else \symplinenum\fi
1391   \ifl@d@ssub \fullstop \sublinenumr@p{#3}\fi
1392   \ifl@d@dash \endashchar\fi
1393   \ifl@d@pnum #4\fullstop\fi
1394   \ifl@d@elin \linenumr@p{#5}\Rlineflag\fi
1395   \ifl@d@esl \ifl@d@elin \fullstop\fi \sublinenumr@p{#6}\fi
1396 \endgroup}
1397
1398 \let\ledsavedprintlines\printlines
1399

```

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.

```

1400 \list@create{\labelref@listR}
1401

```

`\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.

```

1402 \renewcommand*{\edlabel}[1]{%
1403   \ifl@d@pairing\ifautopar%
1404     \strut%
1405   \fi\fi%
1406   \@bsphack%
1407   \ifledRcol

```

```

1408 \write\linenum@outR{\string\@lab}%
1409 \ifx\labelref@listR\empty
1410 \xdef\label@refs{\zz@@@}%
1411 \else
1412 \gl@p\labelref@listR\to\label@refs
1413 \fi
1414 \ifvmode
1415 \advancelabel@refs
1416 \fi
1417 \protected@write\@auxout{%
1418 {\string\l@dmake@labelsR\space\thepage|\label@refs|\the\c@pstartR|{#1}}%
1419 \else
1420 \write\linenum@out{\string\@lab}%
1421 \ifx\labelref@list\empty
1422 \xdef\label@refs{\zz@@@}%
1423 \else
1424 \gl@p\labelref@list\to\label@refs
1425 \fi
1426 \ifvmode
1427 \advancelabel@refs
1428 \fi
1429 \protected@write\@auxout{%
1430 {\string\l@dmake@labelsR\space\thepage|\label@refs|\the\c@pstart|{#1}}%
1431 \fi
1432 \@esphack}
1433
1434

```

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

```

1435 \def\l@dmake@labelsR#1|#2|#3|#4|#5{%
1436 \expandafter\ifx\csname the@label#5\endcsname \relax\else
1437 \led@warn@DuplicateLabel{#4}%
1438 \fi
1439 \expandafter\gdef\csname the@label#5\endcsname{#1|#2\Rlineflag|#3|#4}%
1440 \ignorespaces}
1441 \AtBeginDocument{%
1442 \def\l@dmake@labelsR#1|#2|#3|#4|#5{%
1443 }
1444

```

`\@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.

```

1445 \renewcommand*{\@lab}{%
1446 \ifledRcol
1447 \xright@appenditem{\linenumr@p{\line@numR}|%
1448 \ifsublines@ \sublinenumr@p{\subline@numR}\else 0\fi}%
1449 \to\labelref@listR

```

```

1450 \else
1451   \xright@appenditem{\linenumr@p{\line@num}}|{%
1452     \ifsublines@ \sublinenumr@p{\subline@num}\else 0\fi}%
1453   \to\labelref@list
1454 \fi}
1455

```

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* 1456 \WithSuffix\newcommand\sidenotemargin*[1]{%
1457   \l@dgetsidenote@margin{#1}
1458   \global\sidenote@marginR=\@l@tempcntb
1459   \global\sidenote@margin=\@l@tempcntb
1460 }
1461 \newcount\sidenote@marginR
1462 \global\sidenote@margin=\@ne
1463

```

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

```

1464 \newcommand*\affixside@noteR{%
1465   \def\sidenotecontent@{%
1466     \numgdef{\itemcount@}{0}%
1467     \def\do##1{%
1468       \ifnumequal{\itemcount@}{0}%
1469       {%
1470         \appto\sidenotecontent@{##1}}% Not print not separator before the 1st note
1471       {\appto\sidenotecontent@{\sidenotesep ##1}%
1472       }%
1473       \numgdef{\itemcount@}{\itemcount@+1}%
1474     }%
1475     \dolistloop{\l@dcsofnotetext}%
1476     \ifnumgreater{\itemcount@}{1}{\led@err@ManySidenotes}{}%
1477   \gdef\@templ@d{%
1478     \gdef\@templ@n{\l@dcsofnotetext\l@dcsofnotetext@l\l@dcsofnotetext@r}%
1479     \ifx\@templ@d\@templ@n \else%
1480       \if@twocolumn%
1481         \if@firstcolumn%
1482           \setl@dlp@rbox{##1}{\sidenotecontent@}%
1483         \else%
1484           \setl@drp@rbox{\sidenotecontent@}%
1485         \fi%
1486       \else%
1487         \l@tempcntb=\sidenote@marginR%
1488         \ifnum\l@tempcntb>\@ne%

```



```

1489     \advance\@l@tempcntb by\page@numR%
1490     \fi%
1491     \ifodd\@l@tempcntb%
1492         \setl@drp@rbox{\sidenotecontent@}%
1493         \gdef\sidenotecontent@{}%
1494         \numdef{\itemcount@}{0}%
1495         \dolistloop{\l@dcstetext@l}%
1496         \ifnumgreater{\itemcount@}{1}{\led@err@ManyLeftnotes}{}%
1497         \setl@dlp@rbox{\sidenotecontent@}%
1498     \else%
1499         \setl@dlp@rbox{\sidenotecontent@}%
1500         \gdef\sidenotecontent@{}%
1501         \numdef{\itemcount@}{0}%
1502         \dolistloop{\l@dcstetext@r}%
1503         \ifnumgreater{\itemcount@}{1}{\led@err@ManyRightnotes}{}%
1504         \setl@drp@rbox{\sidenotecontent@}%
1505     \fi%
1506 \fi%
1507 \fi%
1508 }
1509

```

21 Familiar footnotes

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

```

1510 \renewcommand{\l@dbfnote}[1]{%
1511     \ifnumberedpar@
1512     \gdef\@tag{#1}%
1513     \ifledRcol%
1514         \xright@appenditem{\noexpand\vl@dbfnote{\expandonce\@tag}}{\@thefnmark}}%
1515         \to\inserts@listR
1516         \global\advance\insert@countR \@ne%
1517     \else%
1518         \xright@appenditem{\noexpand\vl@dbfnote{\expandonce\@tag}}{\@thefnmark}}%
1519         \to\inserts@list
1520         \global\advance\insert@count \@ne%
1521     \fi
1522 \fi\ignorespaces}
1523

```

`\normalbfnoteX`

```

1524 \renewcommand{\normalbfnoteX}[2]{%
1525     \ifnumberedpar@
1526     \ifledRcol%
1527     \ifluatex
1528         \footnotelang@lua[R]%
1529     \fi

```

```

1530 \ifundefined{xpg@main@language}%if polyglossia
1531 {}%
1532 {\footnotelang@poly[R]}%
1533 \protected@xdef\thisfootnote{\csuse{thefootnote#1}}%
1534 \xright@appenditem{\noexpand\vbfnoteX{#1}{#2}}{\expandonce\thisfootnote}}%
1535 \to\inserts@listR
1536 \global\advance\insert@countR \@ne%
1537 \else%
1538 \ifluatex
1539 \footnotelang@lua%
1540 \fi
1541 \ifundefined{xpg@main@language}%if polyglossia
1542 {}%
1543 {\footnotelang@poly}%
1544 \protected@xdef\thisfootnote{\csuse{thefootnote#1}}%
1545 \xright@appenditem{\noexpand\vbfnoteX{#1}{#2}}{\expandonce\thisfootnote}}%
1546 \to\inserts@list
1547 \global\advance\insert@count \@ne%
1548 \fi
1549 \fi\ignorespaces}
1550

```

22 Verse

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

```

\inserthangingsymbolL
\inserthangingsymbolR 1551 \newif\ifinserthangingsymbolR
1552 \newcommand{\inserthangingsymbolL}{%
1553 \ifinserthangingsymbol%
1554 \ifinstanzaL%
1555 \hangingsymbol%
1556 \fi%
1557 \fi}
1558 \newcommand{\inserthangingsymbolR}{%
1559 \ifinserthangingsymbolR%
1560 \ifinstanzaR%
1561 \hangingsymbol%
1562 \fi%
1563 \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 1564 \newcommand{\correcthangingL}{%

```

```

1565 \ifl@dpaging\else%
1566   \ifinstanzaL%
1567     \ifinserthangingsymbol%
1568       \hskip \@ifundefined{sza@00}{0}{\expandafter%
1569         \noexpand\csname sza@00\endcsname}\stanzaindentbase%
1570     \fi%
1571   \fi%
1572 \fi}
1573
1574 \newcommand{\correcthangingR}{%
1575 \ifl@dpaging\else%
1576   \ifinstanzaR%
1577     \ifinserthangingsymbolR%
1578       \hskip \@ifundefined{sza@00}{0}{\expandafter%
1579         \noexpand\csname sza@00\endcsname}\stanzaindentbase%
1580     \fi%
1581   \fi%
1582 \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.

```

1583 \chardef\next=\catcode'\&
1584 \catcode'\&=\active
1585

```

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

```

1586 \newenvironment{astanza}{%
1587   \startstanzahook
1588   \catcode'\&\active
1589   \global\stanza@count\@ne\stanza@modulo\@ne
1590   \ifnum\usernamecount{sza@00}=\z@
1591     \let\stanza@hang\relax
1592     \let\endlock\relax
1593   \else
1594   %%% \interlinepenalty\@M % this screws things up, but I don't know why
1595     \rightskip\z@ plus 1fil\relax
1596   \fi
1597   \ifnum\usernamecount{szp@00}=\z@
1598     \let\sza@penalty\relax
1599   \fi
1600   \def&{%
1601     \endlock\mbox{}%
1602     \sza@penalty
1603     \global\advance\stanza@count\@ne
1604     \@astanza@line}%
1605   \def\&{%
1606     \endlock\mbox{}

```

```

1607 \pend
1608 \endstanzaextra}%
1609 \pstart
1610 \@astanza@line
1611 }{}
1612

```

`\@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.

```

1613 \newcommand*{\@astanza@line}{%
1614 \ifnum\value{stanzaindentrepetition}=0
1615 \parindent=\csname sza@\number\stanza@count
1616 @\endcsname\stanzaindentbase
1617 \else
1618 \parindent=\csname sza@\number\stanza@modulo
1619 @\endcsname\stanzaindentbase
1620 \managestanza@modulo
1621 \fi
1622 \par
1623 \stanza@hang%\mbox{}%
1624 \ignorespaces}
1625

```

Lastly reset the modified category codes.

```

1626 \catcode'\&=\next
1627

```

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 the regular box macros, but including the string ‘name’.

```

\unhnamebox 1628 \providecommand*{\newnamebox}[1]{%
\unvnamebox 1629 \expandafter\newbox\csname #1\endcsname}
\namebox 1630 \providecommand*{\setnamebox}[1]{%
1631 \expandafter\setbox\csname #1\endcsname}
1632 \providecommand*{\unhnamebox}[1]{%
1633 \expandafter\unhbox\csname #1\endcsname}
1634 \providecommand*{\unvnamebox}[1]{%
1635 \expandafter\unvbox\csname #1\endcsname}
1636 \providecommand*{\namebox}[1]{%
1637 \csname #1\endcsname}
1638

```

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

`\usenamecount`

```

1639 \providecommand*\newnamecount}[1]{%
1640   \expandafter\newcount\csname #1\endcsname}
1641 \providecommand*\usernamecount}[1]{%
1642   \csname #1\endcsname}
1643

```

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.

```

1644 \newcount\l@dc@maxchunks
1645 \newcommand*\maxchunks}[1]{\l@dc@maxchunks=#1}
1646   \maxchunks{5120}
1647

```

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

```

\l@dnumpstartsR 1648 \newcount\l@dnumpstartsR
1649

```

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

```

\l@pscl 1650 \newcount\l@dpscl
1651 \newcount\l@dpsclR
1652

```

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

```

1653 \newcommand*\l@dsetuprawboxes{%
1654   \@l@dttempcntb=\l@dc@maxchunks
1655   \loop\ifnum\@l@dttempcntb>\z@
1656     \newnamebox{\l@dLcolrawbox\the\@l@dttempcntb}
1657     \newnamebox{\l@dRcolrawbox\the\@l@dttempcntb}
1658     \advance\@l@dttempcntb \m@ne
1659   \repeat}
1660

```

`\l@dsetupmaxlinecounts` To be able to synchronise left and right texts we need to know the maximum number 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.

```

1661 \newcommand*\l@dsetupmaxlinecounts{%
1662   \@l@dttempcntb=\l@dc@maxchunks
1663   \loop\ifnum\@l@dttempcntb>\z@

```

```

1664     \newnamecount{l@dmaxlinesinpar\the\@l@tempcntb}
1665     \advance\@l@tempcntb \m@ne
1666     \repeat}
1667 \newcommand*{\l@dzeromaxlinecounts}{%
1668     \begingroup
1669     \@l@tempcntb=\l@dc@maxchunks
1670     \loop\ifnum\@l@tempcntb>\z@
1671         \global\usenamecount{l@dmaxlinesinpar\the\@l@tempcntb}=\z@
1672         \advance\@l@tempcntb \m@ne
1673     \repeat
1674     \endgroup}
1675

```

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

```

1676 \AtBeginDocument{%
1677     \l@dsetupprawboxes
1678     \l@dsetupmaxlinecounts
1679     \l@dzeromaxlinecounts
1680     \l@dnumpestartsL=\z@
1681     \l@dnumpestartsR=\z@
1682     \l@dpscL=\z@
1683     \l@dpscR=\z@}
1684

```

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 1685 \newif\ifl@dusedbabel
\l@dusedbabeltrue 1686 \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 `\`.

```

1687 \newcommand*{\l@dbbl@set@language}[1]{%
1688   \edef\language{#1}%
1689   \select@language{\language}%
1690   \if@filesw
1691     \protected@write\@auxout{}\string\select@language{\language}}%
1692   \addtocontents{toc}{\string\select@language{\language}}%
1693   \addtocontents{lof}{\string\select@language{\language}}%
1694   \addtocontents{lot}{\string\select@language{\language}}%
1695   \fi}
1696

```

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 1697 \providecommand{\selectlanguage}[1]{%
1698   \newcommand*{\l@duselanguage}[1]{%
1699     \gdef\theledlanguageL{}
1700     \gdef\theledlanguageR{}
1701

```

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

```

1702 \AtBeginDocument{%
1703   \@ifundefined{xpg@main@language}{%
1704     \@ifundefined{bbl@main@language}{%

```

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

```

1705   \l@dusedbabelfalse
1706   \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.

```

1707   \l@dusedbabeltrue
1708   \let\l@doldselectlanguage\selectlanguage
1709   \let\l@doldbbl@set@language\bbl@set@language
1710   \let\bbl@set@language\l@doldbbl@set@language
1711   \renewcommand{\selectlanguage}[1]{%
1712     \l@doldselectlanguage{#1}%
1713     \ifledRcol \gdef\theledlanguageR{#1}%
1714     \else      \gdef\theledlanguageL{#1}%
1715     \fi}

```

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

```
1716 \renewcommand*{\l@duselanguage}[1]{%
1717 \l@doldselectlanguage{#1}}
```

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

```
1718 \gdef\theledlanguageL{\bbl@main@language}%
1719 \gdef\theledlanguageR{\bbl@main@language}%
1720 }%
1721 }
```

If on Polyglossia

```
1722 { \let\old@otherlanguage\otherlanguage%
1723 \renewcommand{\otherlanguage}[2][]{%
1724 \selectlanguage[#1]{#2}%
1725 \ifledRcol \gdef\theledlanguageR{#2}%
1726 \else \gdef\theledlanguageL{#2}%
1727 \fi}%
1728 \let\l@duselanguage\select@language%
1729 \gdef\theledlanguageL{\xpg@main@language}%
1730 \gdef\theledlanguageR{\xpg@main@language}%
1731 }}
```

That's it.

```
1731 }}
```

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

```
\@pstartstrue 1732 \newif\if@pstarts
\@pstartsfalse 1733 \newcommand*{\check@pstarts}{%
\check@pstarts 1734 \@pstartsfalse
1735 \ifnum\l@dnumpstartsL>\l@dpscL
1736 \@pstartstrue
1737 \else
1738 \ifnum\l@dnumpstartsR>\l@dpscR
1739 \@pstartstrue
1740 \fi
1741 \fi
1742 }
1743
```

`\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 1744 \newif\ifaraw@text
1745 \araw@textfalse
1746 \newcommand*{\checkraw@text}{%
1747 \araw@textfalse
1748 \ifvbox\namebox{\l@dLcolrawbox\the\l@dpscL}
1749 \araw@texttrue
1750 \else
1751 \ifvbox\namebox{\l@dRcolrawbox\the\l@dpscR}
```



```

1752     \araw@texttrue
1753     \fi
1754 \fi
1755 }
1756

```

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

```

1757 \newcommand*{\@writelinesinparL}{%
1758   \edef\next{%
1759     \write\linenum@out{\string\@pend[\the\@donereallinesL]}}%
1760   \next
1761   \global\@donereallinesL \z@}
1762 \newcommand*{\@writelinesinparR}{%
1763   \edef\next{%
1764     \write\linenum@outR{\string\@pendR[\the\@donereallinesR]}}%
1765   \next
1766   \global\@donereallinesR \z@}
1767

```

26 Parallel columns

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

```

\@eledsectionR 1768 \newsavebox{\@eledsectionL}%
1769 \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.

```

1770 \newcommand*{\Columns}{%
1771   \eledsection@correcting@skip=-\baselineskip% Correction for sections' titles
1772   \setcounter{pstartL}{\value{pstartLold}}
1773   \setcounter{pstartR}{\value{pstartRold}}
1774   \ifnum\l@dnumpstartsL=\l@dnumpstartsR\else
1775     \led@err@BadLeftRightPstarts{\the\l@dnumpstartsL}{\the\l@dnumpstartsR}%
1776   \fi

```

Start a group and zero counters, etc.

```

1777   \begingroup
1778     \l@dzeropenalties
1779     \endgraf\global\num@lines=\prevgraf
1780     \global\num@linesR=\prevgraf
1781     \global\par@line=\z@
1782     \global\par@lineR=\z@
1783     \global\l@dpscL=\z@
1784     \global\l@dpscR=\z@

```

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

```
1785 \check@pstarts
1786 \loop\if@pstarts
1787 \global\pstartnumtrue
1788 \global\pstartnumRtrue
```

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

```
1789 \global\advance\l@dpscL \@ne
1790 \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

```
1791 \checkraw@text
1792 { \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.

```
1793 \l@duselanguage{\theledlanguageL}%
1794 \do@lineL
1795 \xifinlist{\the\l@dpscL}{\eled@sections@@}
1796 {%
1797 \ifdefstring{\@eledsectmark}{L}%
1798 {\csuse{eled@sectmark@the\l@dpscL}%
1799 }{}}%
1800 \global\csundef{eled@sectmark@the\l@dpscL}%
1801 \savebox{\@eledsectionL}{\parbox[t]{}[t]{\Lcolwidth}{\vbox{} \print@eledse
1802 }%
1803 }%
1804 \l@duselanguage{\theledlanguageR}%
1805 \do@lineR
1806 \xifinlist{\the\l@dpscR}{\eled@sectionsR@@}
1807 {%
1808 \ifdefstring{\@eledsectmark}{R}%
1809 {\csuse{eled@sectmark@the\l@dpscR R}%
1810 }{}}%
1811 \global\csundef{eled@sectmark@the\l@dpscR R}%
1812 \savebox{\@eledsectionR}{\parbox[t]{}[t]{\Rcolwidth}{\vbox{} \print@eledse
1813 }%
1814 }%
1815 \hb@xt@ \hsize{%
1816 \ifdefstring{\columns@position}{L}{\hfill }%
1817 \unhbox\l@dleftbox%
1818 \ifhbox\@eledsectionL%
1819 \usebox{\@eledsectionL}%
1820 \fi%
1821 \print@columnseparator%
1822 \unhbox\l@drightbox%
1823 \ifhbox\@eledsectionR%
1824 \usebox{\@eledsectionR}%
1825 \fi%
1826 }
```

```

1824         \fi%
1825         \ifdefstring{\columns@position}{R}{\hfill}%
1826     }%
1827     \checkraw@text
1828     \checkverseL
1829     \checkverseR
1830     \checkpb@columns
1831 \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.

```

1832     \@writelinesinparL
1833     \@writelinesinparR
1834     \check@pstarts
1835     \ifbypstart@
1836         \write\linenum@out{\string\@set[1]}
1837         \resetprevline@
1838     \fi
1839     \ifbypstart@R
1840         \write\linenum@outR{\string\@set[1]}
1841         \resetprevline@
1842     \fi
1843     \addtocounter{pstartL}{1}
1844     \addtocounter{pstartR}{1}
1845 \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.

```

1846     \flush@notes
1847     \flush@notesR
1848 \endgroup
1849 \global\l@dpscL=\z@
1850 \global\l@dpscR=\z@
1851 \global\l@dnumpstartsL=\z@
1852 \global\l@dnumpstartsR=\z@
1853 \ignorespaces
1854 \global\instanzaLfalse
1855 \global\instanzaRfalse}
1856

```

`\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).

```

1857 \def\print@columnseparator{%
1858     \ifdim\beforecolumnseparator<0pt%
1859         \hfill%
1860     \else%
1861         \hspace{\beforecolumnseparator}%
1862     \fi%

```

```

1863 \columnseparator%
1864 \ifdim\aftercolumnseparator<0pt%
1865   \hfill%
1866 \else%
1867   \hspace{\beforecolumnseparator}%
1868 \fi%
1869 }%
1870 %\end{macrocode}
1871 % \end{macro}
1872 % \begin{macro}{\checkpb@columns}
1873 % \cs{checkpb@columns} prevent or make pagebreaking in columns, depending of the use of \
1874 %   \begin{macrocode}
1875
1876 \newcommand{\checkpb@columns}{%
1877   \newif\if@pb
1878   \newif\if@nopb
1879   \IfStrEq{\led@pb@setting}{before}{%
1880     \numdef{\next@absline}{\the\absline@num+1}%
1881     \numdef{\next@abslineR}{\the\absline@numR+1}%
1882     \xifinlistcs{\next@absline}{l@prev@pb}{\@pbtrue}{}%
1883     \xifinlistcs{\next@abslineR}{l@prev@pbR}{\@pbtrue}{%
1884     \xifinlistcs{\next@absline}{l@prev@nopb}{\@nopbtrue}{}%
1885     \xifinlistcs{\next@abslineR}{l@prev@nopbR}{\@nopbtrue}{%
1886   }{}
1887   \IfStrEq{\led@pb@setting}{after}{%
1888     \xifinlistcs{\the\absline@num}{l@prev@pb}{\@pbtrue}{}%
1889     \xifinlistcs{\the\absline@numR}{l@prev@pbR}{\@pbtrue}{%
1890     \xifinlistcs{\the\absline@num}{l@prev@nopb}{\@nopbtrue}{}%
1891     \xifinlistcs{\the\absline@numR}{l@prev@nopbR}{\@nopbtrue}{%
1892   }{}
1893 \if@nopb\nopagebreak[4]\enlargethispage{\baselineskip}\fi
1894 \if@pb\pagebreak[4]\fi
1895 }

```

`\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).

```

1896 \newcommand*{\columnseparator}{%
1897   \smash{\rule[-0.2\baselineskip]{\columnrulewidth}{1.05\baselineskip}}}
1898 \newdimen\columnrulewidth
1899 \columnrulewidth=\z@
1900

```

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

```

1901 \newcommand*{\columnspostion}[1]{%
1902   \xdef\columns@position{#1}%
1903 }%

```

```

1904 \xdef\columns@position{R}%

\beforecolumnseparator \beforecolumnseparator and \aftercolumnseparator lengths are defined to
\aftercolumnseparator -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.

1905 \newlength{\beforecolumnseparator}%
1906 \setlength{\beforecolumnseparator}{-2pt}%
1907
1908 \newlength{\aftercolumnseparator}%
1909 \setlength{\aftercolumnseparator}{-2pt}%
1910

setwidthliketwocolumns@L The \setwidth... macros are called in \beginnumbering in a non-parallel
tpositionliketwocolumns@L typesetting context, to fix the width of the lines to be vertically aligned with par-
eositionliketwocolumns@L allel columns. They are also called at the beginning of a note's group, if some op-
setwidthliketwocolumns@C tions are enabled. The \setposition... macros are called in \beginnumbering
tpositionliketwocolumns@C in a non- parallel typesetting context to fix the position of the lines. The
eositionliketwocolumns@C \setnoteposition... macros are called in \xxxfootstart in a non- paral-
setwidthliketwocolumns@R lel typesetting context to fix the position of notes block.

tpositionliketwocolumns@R 1911 \newcommand{\setwidthliketwocolumns@L}{%
eositionliketwocolumns@R 1912 % Temporary dimension, initially equal to the standard hsize, i.e. text width
1913 % \begin{macrocode}
1914 \newdimen\temp%
1915 \temp=\hsize%

Hsize : Left + Right width
1916 \hsize=\Lcolwidth%
1917 \advance\hsize\Rcolwidth%

Now, calculating the remaining space
1918 \advance\temp-\hsize%

And multiply the hsize by 2/3 of this space
1919 \multiply\temp by 2%
1920 \divide\temp by 3%
1921 \advance\hsize\temp%
1922 }%
1923
1924 \newcommand{\setpositionliketwocolumns@L}{%
1925 \renewcommand{\ledrlfill}{\hfill}%
1926 }%
1927
1928 \newcommand{\setnotespositionliketwocolumns@L}{%
1929 }%
1930
1931

1932 \newcommand{\setwidthliketwocolumns@C}{%
1933 % Temporary dimension, initially equal to the standard hsize, i.e. text width

```

```

1934 \newdimen\temp%
1935 \temp=\hsize%
1936 % Hsize : Left + Right width
1937 \hsize=\Lcolwidth%
1938 \advance\hsize\Rcolwidth%
1939 % Now, calculating the remaining space
1940 \advance\temp-\hsize%

    And multiply the hsize by 1/2 of this space
1941 \divide\temp by 2%
1942 \advance\hsize\temp%
1943 }%
1944
1945 \newcommand{\setpositionliketwocolumns@C}{%
1946 \doinsidelinehook{\hfill}%
1947 \renewcommand{\ledrlfill}{\hfill}%
1948 }%
1949
1950 \newcommand{\setnotespositionliketwocolumns@C}{%
1951 \newdimen\temp%
1952 \newdimen\tempa%
1953 \temp=\hsize%
1954 \tempa=\Lcolwidth%
1955 \advance\tempa\Rcolwidth%
1956 \advance\temp-\tempa%
1957 \divide\temp by 2%
1958 \leftskip=\temp%
1959 \rightskip=-\temp%
1960 }%
1961
1962 \newcommand{\setwidthliketwocolumns@R}{%

    Temporary dimension, initially equal to the standard hsize, i.e. text width
1963 \newdimen\temp%
1964 \temp=\hsize%

    Hsize : Left + Right width
1965 \hsize=\Lcolwidth%
1966 \advance\hsize\Rcolwidth%

    Now, calculating the remaining space
1967 \advance\temp-\hsize%

    And multiply the hsize by 2/3 of this space
1968 \multiply\temp by 2%
1969 \divide\temp by 3%
1970 \advance\hsize\temp%
1971 }%
1972
1973 \newcommand{\setpositionliketwocolumns@R}{%
1974 \doinsidelinehook{\hfill}%

```

```

1975 }%
1976
1977 \newcommand{\setnotespositionliketwocolumns@R}{%
1978   \newdimen\temp%
1979   \newdimen\tempa%
1980   \temp=\hsize%
1981   \tempa=\Lcolwidth%
1982   \advance\tempa\Rcolwidth%
1983   \advance\temp-\tempa%
1984   \divide\temp by 2%
1985   \leftskip=\temp%
1986   \rightskip=-\temp%
1987 }%
1988

```

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 1989 \newcount\numpagelinesL
                  1990 \newcount\numpagelinesR
                  1991 \newcount\l@dminpagelines
                  1992

```

\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.

```

1993 \newcommand*{\Pages}{%
1994   \eledsection@correcting@skip=-2\baselineskip% line correcting for section titles.
1995   \setcounter{pstartL}{\value{pstartLold}}
1996   \setcounter{pstartR}{\value{pstartRold}}
1997   \parledgroup@notespacing@set@correction
1998   \typeout{}
1999   \typeout{***** PAGES *****}
2000   \ifnum\l@dnumpstartsL=\l@dnumpstartsR\else
2001     \led@err@BadLeftRightPstarts{\the\l@dnumpstartsL}{\the\l@dnumpstartsR}%
2002   \fi

```

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

```

2003 \cleartol@devenpage
2004 \begingroup
2005   \l@dzeropenalties
2006   \endgraf\global\num@lines=\prevgraf
2007   \global\num@linesR=\prevgraf
2008   \global\par@line=\z@
2009   \global\par@lineR=\z@

```

```

2010    \global\l@dpscL=\z@
2011    \global\l@dpscR=\z@
2012    \writtenlinesLfalse
2013    \writtenlinesRfalse

```

Check if there are chunks to be processed.

```

2014    \check@pstarts
2015    \loop\if@pstarts

```

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

```

2016    \global\advance\l@dpscL \@ne
2017    \global\advance\l@dpscR \@ne

```

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

```

2018    \getlinesfromparlistL
2019    \getlinesfromparlistR
2020    \l@dcalc@maxoftwo{\@cs@linesinparL}{\@cs@linesinparR}%
2021    {\usernamecount{l@dmmaxlinesinpar\the\l@dpscL}}%
2022    \check@pstarts
2023    \repeat

```

Zero the counts again, ready for the next bit.

```

2024    \global\l@dpscL=\z@
2025    \global\l@dpscR=\z@

```

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

```

2026    \getlinesfrompagelistL
2027    \getlinesfrompagelistR
2028    \l@dcalc@minoftwo{\@cs@linesonpageL}{\@cs@linesonpageR}%
2029    {\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.

```

2030    \check@pstarts
2031    \if@pstarts

```

Increment the chunk counts to get the first pair.

```

2032    \global\advance\l@dpscL \@ne
2033    \global\advance\l@dpscR \@ne

```

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

```

2034    \global\@donereallinesL=\z@
2035    \global\@donetotallinesL=\z@
2036    \global\@donereallinesR=\z@
2037    \global\@donetotallinesR=\z@

```

Start a loop over the boxes (chunks).

```

2038    \checkraw@text

```



```

2039 %      \begingroup
2040 {      \loop\ifaraw@text

```

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

```

2041      \checkpageL
2042      \l@duselanguage{\theledlanguageL}%
2043 %%%      \begingroup
2044 {      \loop\ifl@dsamepage%
    Process the next (left) text line, adding it to the page. Eventually, adds the
    optional argument of pstart.
2045      \ifdefstring{\@eledsectnotoc}{L}{\ledsectnotoc}{%
2046      \csuse{before@pstartL@the\l@dpscl}%
2047      \global\csundef{before@pstartL@the\l@dpscl}
2048      \do@lineL
2049      \xifinlist{\the\l@dpscl}{\eled@sections@@}
2050      {\print@eledsectionL}%
2051      {%
2052      \advance\numpagelinesL \@ne
2053      \ifshiftedpstarts
2054          \ifdim\ht\l@dleftbox>0pt\hb@xt@ \hsize{\ledstrutL\unhbox\l@dleftbox}\fi%
2055      \else%
2056          \parledgroup@correction@notespacing{L}
2057          \hb@xt@ \hsize{\ledstrutL\unhbox\l@dleftbox}%
2058      \fi

```

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

```

2059      \get@nextboxL%
2060      \ifprint@last@after@pendL%
2061      \csuse{after@pendL@the\l@dpscl}%
2062      \global\csundef{after@pendL@the\l@dpscl}%
2063      \fi%
2064      \checkpageL%
2065      \checkverseL
2066      \checkpbl
2067      \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.

```

2068      \ifl@dpagfull
2069      \@writelinesonpageL{\the\numpagelinesL}%
2070      \else
2071      \@writelinesonpageL{1000}%
2072      \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.

```

2073      \numpagelinesL \z@
2074      \parledgroup@correction@notespacing@init
2075      \clearl@dleftpage }%

```

Now do the same for the right text.

```

2076      \checkpageR
2077      \l@duselanguage{\theledlanguageR}%
2078 {
2079      \loop\ifl@dsamepage%
2080      \initnumbering@sectcountR
2081      \ifdefstring{\@eledsectnotoc}{R}{\ledsectnotoc}{%
2082      \csuse{before@pstartR@the\l@dpscR}%
2083      \global\csundef{before@pstartR@the\l@dpscR}
2084      \do@lineR
2085      \xifinlist{\the\l@dpscR}{\eled@sectionsR@@}
2086      {\print@eledsectionR}%
2087      {}%
2088      \advance\numpagelinesR \@ne
2089      \ifshiftedpstarts
2090      \ifdim\ht\l@drightbox>0pt\hb@xt@ \hsize{\ledstrutR\unhbox\l@drightbox}%
2091      \else%
2092      \parledgroup@correction@notespacing{R}
2093      \hb@xt@ \hsize{\ledstrutR\unhbox\l@drightbox}%
2094      \fi
2095      \get@nextboxR%
2096      \ifprint@last@after@pendR%
2097      \csuse{after@pendR@the\l@dpscR}%
2098      \global\csundef{after@pendR@the\l@dpscR}%
2099      \fi%
2100      \checkpageR%
2101      \checkverseR
2102      \checkpbR
2103      \repeat
2104      \ifl@dpagfull
2105      \@writelinesonpageR{\the\numpagelinesR}%
2106      \else
2107      \@writelinesonpageR{1000}%
2108      \fi
2109      \numpagelinesR=\z@
2110      \parledgroup@correction@notespacing@init

```

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

```

2110      \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.

```

2111      \checkraw@text

```

```

2112      \ifaraw@text
2113      \getlinesfrompagelistL
2114      \getlinesfrompagelistR
2115      \l@dcalc@minoftwo{\@cs@linesonpageL}{\@cs@linesonpageR}%
2116      {\l@dminpagelines}%
2117      \fi
2118      \repeat}

```

We have now output the text from all the chunks.

```

2119      \fi
      Make sure that there are no inserts hanging around.
2120      \flush@notes
2121      \flush@notesR
2122      \endgroup

```

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

```

2123      \global\l@dpscL=\z@
2124      \global\l@dpscR=\z@
2125      \global\l@dnumpestartsL=\z@
2126      \global\l@dnumpestartsR=\z@
2127      \global\instanzaLfalse
2128      \global\instanzaRfalse
2129      \ignorespaces}
2130

```

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

```

\ledstrutR 2131 \newcommand*{\ledstrutL}{\strut}
2132 \newcommand*{\ledstrutR}{\strut}
2133

```

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

```

2134 \providecommand{\cleartoevenpage}[1][\@empty]{%
2135   \clearpage
2136   \ifodd\c@page\hbox{#1}\clearpage\fi}
2137 \newcommand*{\cleartol@devenpage}{%
2138   \ifdim\pagetotal<\topskip% on an empty page
2139   \else
2140     \clearpage
2141   \fi
2142   \ifodd\c@page\hbox{#1}\clearpage\fi}
2143 \newcommand*{\clearl@dleftpage}{%
2144   \clearpage
2145   \ifodd\c@page\else
2146     \led@err@LeftOnRightPage

```

```

2147 \hbox{}%
2148 \cleardoublepage
2149 \fi}
2150 \newcommand*{\clearl@drighthouse}{%
2151 \clearpage
2152 \ifodd\c@page
2153 \led@err@RightOnLeftPage
2154 \hbox{}%
2155 \cleartoevenpage
2156 \fi}
2157

```

`\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 2158 \newcommand*{\getlinesfromparlistL}{%
2159 \ifx\linesinpar@listL\empty
2160 \gdef\cs@linesinparL{0}%
2161 \else
2162 \gl@p\linesinpar@listL\to\cs@linesinparL
2163 \fi}
2164 \newcommand*{\getlinesfromparlistR}{%
2165 \ifx\linesinpar@listR\empty
2166 \gdef\cs@linesinparR{0}%
2167 \else
2168 \gl@p\linesinpar@listR\to\cs@linesinparR
2169 \fi}
2170

```

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

```

\cs@linesonpageR 2171 \newcommand*{\getlinesfrompagelistL}{%
2172 \ifx\linesonpage@listL\empty
2173 \gdef\cs@linesonpageL{1000}%
2174 \else
2175 \gl@p\linesonpage@listL\to\cs@linesonpageL
2176 \fi}
2177 \newcommand*{\getlinesfrompagelistR}{%
2178 \ifx\linesonpage@listR\empty
2179 \gdef\cs@linesonpageR{1000}%
2180 \else
2181 \gl@p\linesonpage@listR\to\cs@linesonpageR
2182 \fi}
2183

```

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

```

2184 \newcommand*{\@writelinesonpageL}[1]{%

```

```

2185 \edef\next{\write\linenum@out{\string\@lopL{#1}}}%
2186 \next}
2187 \newcommand*{\@writelinesonpageR}[1]{%
2188 \edef\next{\write\linenum@outR{\string\@lopR{#1}}}%
2189 \next}
2190

\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>.

2191 \newcommand*{\l@dcalc@maxoftwo}[3]{%
2192 \ifnum #2>#1\relax
2193 #3=#2\relax
2194 \else
2195 #3=#1\relax
2196 \fi}
2197 \newcommand*{\l@dcalc@minoftwo}[3]{%
2198 \ifnum #2<#1\relax
2199 #3=#2\relax
2200 \else
2201 #3=#1\relax
2202 \fi}
2203

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

\checkpageL 2204 \newif\ifl@dsamepage
\checkpageR 2205 \l@dsamepagetrue
2206 \newif\ifl@dpagetrue
2207
2208 \newcommand*{\checkpageL}{%
2209 \l@dpagetrue
2210 \l@dsamepagetrue
2211 \check@goal
2212 \ifdim\pagetotal<\ledthegoal
2213 \ifnum\numpagelinesL<\l@dminpagelines
2214 \else
2215 \l@dsamepagefalse
2216 \l@dpagetruefalse
2217 \fi
2218 \else
2219 \l@dsamepagefalse
2220 \l@dpagetrue
2221 \fi%

```

```

2222 \ifprint@last@after@pendL%
2223     \l@dpagfullfalse%
2224     \l@dsamepagefalse%
2225     \print@last@after@pendLfalse%
2226 \fi%
2227 }%
2228
2229 \newcommand*{\checkpageR}{%
2230     \l@dpagfulltrue
2231     \l@dsamepagetrue
2232     \check@goal
2233     \ifdim\pagetotal<\ledthegoal
2234         \ifnum\numpagelinesR<\l@dminpagelines
2235             \else
2236                 \l@dsamepagefalse
2237                 \l@dpagfullfalse
2238             \fi
2239         \else
2240             \l@dsamepagefalse
2241             \l@dpagfulltrue
2242         \fi%
2243     \ifprint@last@after@pendR%
2244         \l@dpagfullfalse%
2245         \l@dsamepagefalse%
2246         \print@last@after@pendRfalse%
2247     \fi%
2248 }%
2249

```

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

```

2250 \newcommand{\checkpbL}{
2251     \IfStrEq{\led@pb@setting}{after}{
2252         \xifinlistcs{\the\absline@num}{\l@prev@pb}{\l@dpagfulltrue\l@dsamepagefalse}{
2253             \xifinlistcs{\the\absline@num}{\l@prev@nopb}{\l@dpagfullfalse\l@dsamepagetrue}{
2254                 {}
2255             }
2256         \IfStrEq{\led@pb@setting}{before}{
2257             \numdef{\next@absline}{\the\absline@num+1}
2258             \xifinlistcs{\next@absline}{\l@prev@pb}{\l@dpagfulltrue\l@dsamepagefalse}{
2259                 \xifinlistcs{\next@absline}{\l@prev@nopb}{\l@dpagfullfalse\l@dsamepagetrue}{
2260                     {}
2261                 }
2262             }
2263         \IfStrEq{\led@pb@setting}{after}{
2264             \xifinlistcs{\the\absline@numR}{\l@prev@pbR}{\l@dpagfulltrue\l@dsamepagefalse}{
2265                 \xifinlistcs{\the\absline@numR}{\l@prev@nopbR}{\l@dpagfullfalse\l@dsamepagetrue}{
2266                     {}
2267                 }
2268             \IfStrEq{\led@pb@setting}{before}{

```

```

2273 \newcommand{\checkverseL}{
2274 \ifinstanzaL
2275 \iflednopbinverse
2276 \ifinserthangingsymbol
2277 \numgdef{\prev@abslineverse}{\the\absline@num-1}
2278 \IfStrEq{\led@pb@setting}{after}{\lednopbnum{\prev@abslineverse}}{}
2279 \IfStrEq{\led@pb@setting}{before}{\ifnum\numpagelinesL<3\ledpbnum{\prev@abslineverse}\fi}{}
2280 \fi
2281 \fi
2282 \fi
2283 }
2284 \newcommand{\checkverseR}{
2285 \ifinstanzaR
2286 \iflednopbinverse
2287 \ifinserthangingsymbolR
2288 \numgdef{\prev@abslineverse}{\the\absline@numR-1}
2289 \IfStrEq{\led@pb@setting}{after}{\lednopbnumR{\prev@abslineverse}}{}
2290 \IfStrEq{\led@pb@setting}{before}{\ifnum\numpagelinesR<3\ledpbnumR{\prev@abslineverse}\fi}{}
2291 \fi
2292 \fi
2293 \fi
2294 }

```

```

2295 \newdimen\ledthegoal
2296 \ifshiftedpstarts
2297     \newcommand*{\goalfraction}{0.95}
2298 \else
2299     \newcommand*{\goalfraction}{0.9}
2300 \fi
2301
2302 \newcommand*{\check@goal}{%
2303     \ledthegoal=\goalfraction\pagegoal}
2304

```

```
\ifwrittenlinesL 2305 \newif\ifwrittenlinesL
2306 \newif\ifwrittenlinesR
2307
```

`\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.

```

2308 \newcommand*{\get@nextboxL}{%
2309   \ifvbox\namebox{1@dLcolrawbox\the\1@dpscL}% box is not empty

   The current box is not empty; do nothing.
2310   \else%                                     box is empty

   The box is empty. Check if enough lines (real and blank) have been output.
2311     \ifnum\usernamecount{1@dmaxlinesinpar\the\1@dpscL}>\@donetotallinesL
2312       \parledgroup@notes@endL
2313     \else

   Sufficient lines have been output.
2314       \ifnum\usernamecount{1@dmaxlinesinpar\the\1@dpscL}=\@donetotallinesL
2315         \parledgroup@notes@endL
2316       \fi
2317       \ifwrittenlinesL\else

   Write out the number of lines done, and set the boolean so this is only done once.
2318         \@writelinesinparL
2319         \writtenlinesLtrue
2320       \fi
2321       \ifnum\1@dnumpstartsL>\1@dpscL

   There are still unprocessed boxes. Recalculate the maximum number of lines
   needed, and move onto the next box (by incrementing \1@dpscL). If needed, restart
   the line numbering. Increment the pstartL counter.
2322         \writtenlinesLfalse
2323         \ifbypstart@
2324           \ifnum\value{pstartL}<\value{pstartLold}
2325           \else
2326             \global\line@num=0
2327             \resetprevline@
2328           \fi
2329         \fi
2330 % Add the content of the optional argument of the previous \cs{pend}.
2331 %   \begin{macrocode}
2332       \csuse{after@pendL@the\1@dpscL}%
2333       \global\csundef{after@pendL@the\1@dpscL}%
2334 %   \end{macrocode}
2335 %   \begin{macrocode}
2336       \addtocounter{pstartL}{1}
2337       \global\pstartnumtrue
2338       \1@dcalc@maxoftwo{\the\usernamecount{1@dmaxlinesinpar\the\1@dpscL}}%
2339                           {\the\@donetotallinesL}%
2340                           {\usernamecount{1@dmaxlinesinpar\the\1@dpscL}}%
2341       \global\@donetotallinesL \z@
2342       \global\advance\1@dpscL \@ne

```

Add notes of parallel ledgroup.


```

2343     \parledgroup@notes@endL
2344     \parledgroup@correction@notes@spacing@final{L}
2345     \else
2346 % Add the content of the optional argument of the last \cs{pend}.
2347 %     \begin{macrocode}
2348     \print@last@after@pendLtrue%
2349 %     \end{macrocode}
2350 %     \begin{macrocode}
2351     \fi
2352 \fi
2353 \fi}

2354 \newcommand*{\get@nextboxR}{%
2355   \ifvbox\namebox{\l@dRcolrawbox\the\l@dpscR}% box is not empty
2356   \else% box is empty
2357     \ifnum\usenamecount{\l@dmaxlinesinpar\the\l@dpscR}>\@donetotallinesR
2358       \parledgroup@notes@endR
2359     \else
2360       \ifnum\usenamecount{\l@dmaxlinesinpar\the\l@dpscR}=\@donetotallinesR
2361         \parledgroup@notes@endR
2362       \fi
2363       \ifwrittenlinesR\else
2364         \@writelinesinparR
2365         \writtenlinesRtrue
2366       \fi
2367       \ifnum\l@dnumpstartsR>\l@dpscR
2368         \writtenlinesRfalse
2369         \ifbypstartR
2370           \ifnum\value{pstartR}<\value{pstartRold}
2371             \else
2372               \global\line@numR=0
2373               \resetprevline@
2374             \fi
2375           \fi
2376           \csuse{after@pendR@the\l@dpscR}%
2377           \global\csundef{after@pendR@the\l@dpscR}%
2378           \addtocounter{pstartR}{1}
2379           \global\pstartnumRtrue
2380           \l@dcalc@maxoftwo{\the\usenamecount{\l@dmaxlinesinpar\the\l@dpscR}}%
2381             {\the\@donetotallinesR}%
2382             {\usenamecount{\l@dmaxlinesinpar\the\l@dpscR}}}%
2383           \global\@donetotallinesR \z@
2384           \global\advance\l@dpscR \@ne
2385           \parledgroup@notes@endR
2386           \parledgroup@correction@notes@spacing@final{R}
2387         \else
2388           \print@last@after@pendRtrue%
2389         \fi
2390       \fi
2391     \fi}

```

2392

28 Sections' titles' commands

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

```
2393 \newcommand{\eledsectnotoc}[1]{\xdef\@eledsectnotoc{#1}}
2394 \eledsectnotoc{R}
```

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

```
2395 \newcommand{\eledsectmark}[1]{\xdef\@eledsectmark{#1}}
2396 \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`.

```
2397 \newskip\eledsection@correcting@skip
```

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

```
2398 \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`.

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

```
2399 \def\l@prev@pbR{}
2400 \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.

```
2401 \newcommand{\ledpbR}{\write\linenum@outR{\string\led@pbR}}
2402 \newcommand{\ledpbnumR}[1]{\write\linenum@outR{\string\led@pbnumR{#1}}}
2403 \newcommand{\lednopbR}{\write\linenum@outR{\string\led@nopbR}}
2404 \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` `\led@pbnumR` add the argument in the `\prev@pbR` list. The `\led@nopbR` add
`\led@nopbR` the absolute line number in the `\prev@nopbR` list. The `\led@nopbnumR` add the
`\led@nopbnumR` argument in the `\prev@nopbR` list.

```

2405 \newcommand{\led@pbR}{\listxadd{\l@prev@pbR}{\the\absline@numR}}
2406 \newcommand{\led@pbnumR}[1]{\listxadd{\l@prev@pbR}{\#1}}
2407 \newcommand{\led@nopbR}{\listxadd{\l@prev@nopbR}{\the\absline@numR}}
2408 \newcommand{\led@nopbnumR}[1]{\listxadd{\l@prev@nopbR}{\#1}}

```

30 Parallel ledgroup

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

```

2409 \newmarks\parledgroup@
2410 \newmarks\parledgroup@series
2411 \newmarks\parledgroup@type

```

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

```

2412 \newcommand{\parledgroup@notes@startL}{%
2413   \ifnum\usenamecount{\l@maxlinesinpar\the\l@dpscL}>0%
2414     \IfStrEq{\splitfirstmarks\parledgroup@type}{footnoteX}{\csuse{bhooknoteX@\splitfirstmarks\parledg
2415       \IfStrEq{\splitfirstmarks\parledgroup@type}{Xfootnote}{\csuse{bhookXnote@\splitfirstmarks\parledg
2416       \fi%
2417       \global\ledgroupnotesL=true%
2418       \insert@noterule@ledgroup{L}%
2419   }
2420 \newcommand{\parledgroup@notes@startR}{%
2421   \ifnum\usenamecount{\l@maxlinesinpar\the\l@dpscR}>0%
2422     \IfStrEq{\splitfirstmarks\parledgroup@type}{footnoteX}{\csuse{bhooknoteX@\splitfirstmarks\parledg
2423     \IfStrEq{\splitfirstmarks\parledgroup@type}{Xfootnote}{\csuse{bhookXnote@\splitfirstmarks\parledg
2424     \fi%
2425     \global\ledgroupnotesR=true%
2426     \insert@noterule@ledgroup{R}%
2427   }

```

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

```

2428 \newcommand{\parledgroup@notes@endL}{%
2429   \global\ledgroupnotesL=false%
2430   }
2431 \newcommand{\parledgroup@notes@endR}{%
2432   \global\ledgroupnotesR=false%
2433   }

```

`\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`

```

2434 \newcommand{\insert@noterule@ledgroup}[1]{
2435   \IfStrEq{\splitbotmarks\parledgroup@}{begin}{%
2436     \IfStrEq{\splitbotmarks\parledgroup@type}{Xfootnote}{
2437       \csuse{ifledgroupnotes#1@}
2438       \vskip\skip\csuse{mp\splitbotmarks\parledgroup@series footins}
2439       \csuse{\splitbotmarks\parledgroup@series footnoterule}
2440       \fi
2441     }
2442   {}
2443   \IfStrEq{\splitbotmarks\parledgroup@type}{footnoteX}{
2444     \csuse{ifledgroupnotes#1@}
2445     \vskip\skip\csuse{mpfootins\splitbotmarks\parledgroup@series}
2446     \csuse{footnoterule\splitbotmarks\parledgroup@series}
2447     \fi
2448   }{}
2449 }
2450 {}
2451 }
```

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

```

2452 \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`.

```

2453 \dimdef{\parledgroup@notespacing@correction}{0pt}
2454 \newcommand{\parledgroup@notespacing@set@correction}{%
2455   {\notefontsetup\parledgroupnotespacing\dimgdef{\temp@spacing}{\baselineskip}}%
2456   \dimgdef{\parledgroup@notespacing@correction}{\baselineskip-\temp@spacing}%
2457 }

```

`\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.

```

2458 \newcommand{\parledgroup@correction@notespacing@init}{
2459   \dimdef{\parledgroup@notespacing@correction@accumulated}{0pt}
2460   \dimdef{\parledgroup@notespacing@correction@modulo}{0pt}
2461 }
2462 \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`.

```

2463 \newcommand{\parledgroup@correction@notespacing@final}[1]{
2464   \ifparledgroup
2465   \vspace{\parledgroup@notespacing@correction@accumulated}
2466   \parledgroup@correction@notespacing@init%
2467   \ifstrequal{#1}{L}{
2468     \numdef{\@checking}{\the\l@dpscL-1}
2469   }{
2470     \numdef{\@checking}{\the\l@dpscR-1}
2471   }
2472   \dimdef{\@beforenotes@current@diff}{\csuse{@parledgroup@beforenotes@\@checking L}-\csuse{@parledg
2473   \ifstrequal{#1}{L}%
2474     {% Left
2475     \ifdimgreater{\@beforenotes@current@diff}{0pt}{\vspace{-\@beforenotes@current@diff}}%
2476     }%
2477     {% Right
2478     \ifdimgreater{\@beforenotes@current@diff}{0pt}{\vspace{\@beforenotes@current@diff}}{}}
2479     }%
2480   \fi
2481 }

```

`\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.

```

2482 {}
2483 \newcommand{\parledgroup@correction@notespacing}[1]{%
2484   \csuse{ifledgroupnotes#1@}%
2485   \vspace{-\parledgroup@notespacing@correction}%
2486   \dimdef{\parledgroup@notespacing@correction@accumulated}{\parledgroup@notespacing@correction@ac
2487   \dimdef{\parledgroup@notespacing@correction@modulo}{\parledgroup@notespacing@correction@modulo+
2488   \ifdimless{\parledgroup@notespacing@correction@modulo}{\baselineskip}{\advance\numpagelinesL
2489   \dimdef{\parledgroup@notespacing@correction@modulo}{\parledgroup@notespacing@correction@modulo-
2490   }% mean greater than equal
2491   \fi%
2492 }

```

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

```

2493 \dimdef\parledgroup@beforenotesL{0pt}
2494 \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.

```

2495 \newcommand{\parledgroup@beforenotes@save}[1]{
2496   \ifparledgroup
2497     \csdingdef{@parledgroup@beforenotes@\the\csuse{1@dnumstarts#1}#1}{\csuse{parledgroup
2498       \csdingdef{parledgroup@beforenotes#1}{Opt}
2499   \fi
2500 }
```

31 The End

i/code_i

Appendix A Some things to do when changing version

Appendix A.1 Migration to eledpar 1.4.3

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

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

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

See the two following examples:

With standard `\hangingsymbol`:

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

With the modification of `\hangingsymbol`:

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

References

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

Index

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

Symbols	<code>\@adv</code>	370, 630, 631
<code>\&</code>	1583, 1584, 1588, 1605, 1626	<code>\@afterindentfalse</code> 766
<code>\@M</code>	1594	<code>\@arabic</code> 222, 223, 815, 818

- \@astanza@line 1604, 1610, 1613
 - \@auxout 1417, 1429, 1691
 - \@beforenotes@current@diff
. 2472, 2475, 2478
 - \@chapter 767
 - \@checking 2468, 2470, 2472
 - \@cs@linesinparL 2020, 2158
 - \@cs@linesinparR 2020, 2158
 - \@cs@linesonpageL 2028, 2115, 2171
 - \@cs@linesonpageR 2028, 2115, 2171
 - \@currentlabel 858, 901
 - \@donereallinesL
. 960, 989, 1759, 1761, 2034
 - \@donereallinesR
. 960, 1061, 1764, 1766, 2036
 - \@donetotallinesL 960, 990,
993, 2035, 2311, 2314, 2339, 2341
 - \@donetotallinesR 960, 1062,
1065, 2037, 2357, 2360, 2381, 2383
 - \@edtext@false 688, 714
 - \@edtext@true 666, 692
 - \@eled@sectioningfalse 1013
 - \@eled@sectioningtrue 1011
 - \@eledsectionL 1768, 1801, 1817, 1818
 - \@eledsectionR 1768, 1812, 1822, 1823
 - \@eledsectmark 1008, 1797, 1808, 2395
 - \@eledsectnotoc 1007, 2045, 2080, 2393
 - \@gobble 559
 - \@gobbletwo 564
 - \@insertR 1367–1369, 1382–1384
 - \@l@dttempcnta 443,
445, 447, 448, 452, 454, 456,
457, 1119, 1158, 1159, 1161,
1163, 1166, 1167, 1184–1188,
1190, 1197, 1202, 1206, 1214,
1219, 1223, 1256, 1259, 1261, 1265
 - \@l@dttempcntb 178, 180, 182,
1149, 1150, 1197, 1202, 1206,
1214, 1219, 1223, 1248, 1252,
1265, 1273–1275, 1277, 1298–
1300, 1302, 1319–1321, 1323,
1458, 1459, 1487–1489, 1491,
1654–1658, 1662–1665, 1669–1672
 - \@lab 563, 1408, 1420, 1445
 - \@lemmacommand@false 689, 715
 - \@lock 977, 1099
 - \@lockR 56, 314, 316, 318, 331, 477,
493, 494, 496, 497, 525, 526,
528, 1048, 1081, 1125, 1127,
1128, 1130, 1211, 1228, 1230, 1232
 - \@lopL 588, 2185
 - \@lopR 559, 588, 2188
 - \@nl 292, 613, 615
 - \@nl@reg 341
 - \@nl@regR 292
 - \@nobreakfalse 824, 868, 1010
 - \@nobreaktrue 822, 826, 866, 870, 1010
 - \@nobbtrue 1884, 1885, 1890, 1891
 - \@oldnobreak 822, 824, 866, 868, 919, 940
 - \@pbtrue 1882, 1883, 1888, 1889
 - \@pend 579, 1759
 - \@pendR 579, 1764
 - \@pstartsfalse 1732
 - \@pstartstrue 1732
 - \@ref 549
 - \@ref@reg 577
 - \@schapter 767
 - \@set 402, 637, 638, 1836, 1840
 - \@startstanza 779, 780, 802, 803
 - \@sw 564
 - \@tag 668, 694, 1512, 1514, 1518
 - \@templ@d 1477, 1479
 - \@templ@n 1478, 1479
 - \@writelinesinparL 1757, 1832, 2318
 - \@writelinesinparR 1757, 1833, 2364
 - \@writelinesonpageL 2069, 2071, 2184
 - \@writelinesonpageR 2104, 2106, 2184
 - \@xloop 1380
- A**
- \absline@num
366, 436, 450, 469, 1090, 1880,
1888, 1890, 2252, 2253, 2256, 2277
 - \absline@numR 51, 239, 294,
297, 300, 433, 441, 462, 481,
515, 543, 554, 1072, 1111, 1112,
1149, 1366, 1881, 1889, 1891,
2264, 2265, 2268, 2288, 2405, 2407
 - \actionlines@list
. 284, 287, 436, 450, 469
 - \actionlines@listR
. 243, 260, 276, 279, 433,
441, 462, 481, 515, 543, 1171, 1174
 - \actions@list 288, 437, 457, 471, 473
 - \actions@listR
. 243, 261, 280, 434, 448, 464,
466, 483, 492, 517, 524, 544, 1175
 - \add@inserts 986, 1000
 - \add@inserts@nextR 1355
 - \add@insertsR 1057, 1355

- \add@penaltiesL 988, 1376
 \add@penaltiesR 1060, 1376
 \addtocontents 1692–1694
 \addtocounter 921,
 942, 1006, 1843, 1844, 2336, 2378
 \advancelabel@refs 1415, 1427
 \advanceline 629, 660
 \affixline@num 984
 \affixline@numR 1055, 1181
 \affixpstart@numL 998, 1288
 \affixpstart@numR 1288
 \affixside@note 986, 1000
 \affixside@noteR 1057, 1464
 \aftercolumnseparator . . 6, 1864, 1905
 \appto 1470, 1471
 \araw@textfalse 1744
 \araw@texttrue 1744
 stanza (environment) 10, 1586
 \at@begin@pairs 743, 748, 749
 \at@every@pend 925, 946
 \at@every@pstart 861, 904
 \AtBeginDocument . . . 1441, 1676, 1702
 \AtBeginPairs 5, 748
- B**
- \ballast@count 1109, 1114
 \bbl@main@language 1718, 1719
 \bbl@set@language 1709, 1710
 \beforecolumnseparator
 6, 1858, 1861, 1867, 1905
 \beginnumbering 9, 790, 829
 \beginnumberingR . . . 42, 123, 790, 873
 \bfseries 815, 818
 \bypage@Rfalse 143, 158, 163
 \bypage@Rtrue 143, 153
 \bypstart@Rfalse 143, 154, 164
 \bypstart@Rtrue 143, 159
- C**
- \c@ballast 1114
 \c@chapter 104
 \c@chapterR 104
 \c@firstlinenumR 187, 1254
 \c@firstsublinenumR 191, 1249
 \c@linenumincrementR 187, 1254
 \c@page 613, 615, 2136, 2142, 2145, 2152
 \c@pstart 1430
 \c@pstartL 815
 \c@pstartR 818, 1418
 \c@section 105
 \c@sectionR 105
 \c@sublinenumincrementR . . 191, 1249
 \c@subsection 106
 \c@subsectionR 106
 \c@subsubsection 107
 \c@subsubsectionR 107
 \ch@ck@l@ckR 1181
 \ch@cksub@l@ckR 1181
 \ch@cksub@lockR 1250
 \chapter 752, 753, 762
 \chapterinpages 739, 753, 764
 \chardef 1583
 \check@goal 2211, 2232, 2295
 \check@pstarts
 1732, 1785, 1834, 2014, 2022, 2030
 \checkpageL 2041, 2064, 2204
 \checkpageR 2076, 2099, 2204
 \checkpb@columns . . . 1830, 1872, 1876
 \checkpbL 2066, 2250
 \checkpbR 2101, 2250
 \checkraw@text
 1744, 1791, 1827, 2038, 2111
 \checkverseL 1828, 2065, 2273
 \checkverseR 1829, 2100, 2273
 \cleardoublepage 2148
 \clearl@dleftpage 2075, 2134
 \clearl@drightpage 2110, 2134
 \cleartoevenpage 2134
 \cleartol@devenpage 2003, 2134
 \closeout 95, 601, 608
 \columnrulewidth 6, 1896
 \Columns 5, 1770
 \columns@position . . 1815, 1825, 1901
 \columnseparator 6, 1863, 1896
 \columnsposition 6, 1901
 \correcthangingL 1002, 1564
 \correcthangingR 1564
 \countLline 955, 966
 \countRline 955, 1037
 \critext 665
 \cs 1873, 2330, 2346
 \csdimgdef 2497, 2498
 \csgdef 861,
 862, 904, 905, 925, 926, 946, 947
 \csundef 1014, 1800, 1811,
 2047, 2062, 2082, 2097, 2333, 2377
 \csuse 1012,
 1533, 1544, 1798, 1809, 2046,
 2061, 2081, 2096, 2332, 2376,

2414, 2415, 2422, 2423, 2437–
2439, 2444–2446, 2472, 2484, 2497

D

`\DeclareOption` 8–11
`\def@tempb` 161
`\dimdef` 2453, 2459, 2460,
2472, 2486, 2487, 2489, 2493, 2494
`\dimen` 616, 617, 621–623, 627
`\dingdef` 2455, 2456
`\divide` 1186, 1920, 1941, 1957, 1969, 1984
`\do@actions` 1091
`\do@actions@fixedcodeR` 1118
`\do@actions@nextR` 1118
`\do@actionsR` 1073, 1118
`\do@ballast` 1092
`\do@ballastR` 1074, 1109
`\do@insidelineLhook` . 1002, 1027, 1030
`\do@insidelineRhook` 1028, 1030
`\do@lineL` 965, 1794, 2048
`\do@lineLhook` 970, 1025, 1030
`\do@lineR` 1035, 1805, 2083
`\do@lineRhook` 1026, 1030, 1041
`\do@lockoff` 512
`\do@lockoffL` 536
`\do@lockoffR` 512
`\do@lockon` 477
`\do@lockonL` 509
`\do@lockonR` 477
`\doinsidelinehook` 1946, 1974
`\doinsidelineLhook` 1025
`\doinsidelineRhook` 1025
`\dolineLhook` 1025
`\dolineRhook` 1025
`\dolistloop` 1475, 1495, 1502
`\dummy@ref` 558

E

`\edfont@info` 722, 725, 731, 734
`\edlabel` 1402
`\edtext` 691
`\eled@sectioningR@out` .. 70, 95, 2398
`\eled@sections@@` 985, 1010, 1795, 2049
`\eled@sectionsR@@` 67, 1056, 1806, 2084
`\eledpar@error` 21, 30, 33, 35
`\eledsection@correcting@skip` ..
..... 1022, 1771, 1994, 2397
`\eledsectmark` 14, 2395
`\eledsectnotoc` 14, 2393

`\empty` .. 80, 83, 276, 284, 679, 705,
720, 729, 838, 882, 1171, 1253,
1261, 1357–1359, 1370, 1381,
1409, 1421, 2159, 2165, 2172, 2178
`\end@lemmas` 679, 680, 705, 706
`\endashchar` 1392
`\endgraf` 915, 936, 1779, 2006
`\endline@num` 567, 573
`\endlock` 649, 1592, 1601, 1606
`\endnumbering` 9, 73, 127, 791
`\endnumberingR` 45, 73, 110, 122, 135, 791
`\endpage@num` 566, 573
`\endstanzaextra` 1608
`\endsub` 616
`\endsubline@num` 568, 574
`\enlargethispage` 1893

environments:

`astanza` 10, 1586
`Leftside` 7, 771
`pages` 6, 739
`pairs` 5, 739
`Rightside` 7, 788
`\expandonce` 1514, 1518, 1534, 1545
`\extensionchars` 62, 116, 132, 140

F

`\f@x@l@cksR` 1181
`\first@linenum@out@Rfalse` .. 596, 602
`\first@linenum@out@Rtrue` 596
`\firstlinenum` 7, 196
`\firstlinenum*` 7, 196
`\firstsublinenum` 7, 196
`\firstsublinenum*` 7, 196
`\fix@page` 337, 344
`\flag@end`
..... 616, 675, 685, 686, 701, 711, 712
`\flag@start`
..... 616, 674, 675, 686, 700, 701, 712

`\flush@notes` 1846, 2120
`\flush@notesR` 1379, 1847, 2121
`\footnotelang@lua` 1528, 1539
`\footnotelang@poly` 1532, 1543
`\fullstop` . 235, 1389, 1391, 1393, 1395

G

`\get@linelistfile` 272
`\get@nextboxL` 2059, 2308
`\get@nextboxR` 2094, 2308
`\getline@numL` 976, 1089
`\getline@numR` 1047, 1071

- `\getlinesfrompagelistL` 2026, 2113, 2171
`\getlinesfrompagelistR` 2027, 2114, 2171
`\getlinesfromparlistL` ... 2018, 2158
`\getlinesfromparlistR` ... 2019, 2158
`\gl@p` 279, 280,
 287, 288, 680, 706, 724, 733,
 1174, 1175, 1363, 1367, 1382,
 1412, 1424, 2162, 2168, 2175, 2181
`\goalfraction` 7, 2295
- H**
- `\hangingsymbol` 12, 1555, 1561
`\hb@xt@` 983, 992, 1002, 1054,
 1064, 1814, 2054, 2057, 2089, 2092
`\hsize` 856, 899, 1814, 1915–
 1918, 1921, 1935, 1937, 1938,
 1940, 1942, 1953, 1964–1967,
 1970, 1980, 2054, 2057, 2089, 2092
- I**
- `\if@filesw` 1690
`\if@firstcolumn` 1267, 1292, 1313, 1481
`\if@nobreak` 821, 865
`\if@noeled@sec` 68, 94
`\if@nopb` 1878, 1893
`\if@pb` 1877, 1894
`\if@pstarts` 1732, 1786, 2015, 2031
`\if@RTL` 675, 686, 701, 712, 1015
`\ifaraw@text` ... 1744, 1792, 2040, 2112
`\ifautopar` 848, 892, 1403
`\ifbypage@` 360
`\ifbypage@R` 143, 350, 1153
`\ifbypstart@` 581, 1835, 2323
`\ifbypstart@R` ... 143, 585, 1839, 2369
`\ifdefstring` 1007, 1008,
 1797, 1808, 1815, 1825, 2045, 2080
`\ifdim` ... 617, 621, 623, 627, 1858,
 1864, 2054, 2089, 2138, 2212, 2233
`\ifdimgreater` 2475, 2478
`\ifdimless` 2488
`\iffirst@linenum@out@R` 596, 600
`\ifhbox` 1817, 1822
`\ifinserthangingsymbol`
 1553, 1567, 2276
`\ifinserthangingsymbolR`
 1551, 1559, 1577, 2287
`\ifinstanzaL` 769, 769, 1554, 1566, 2274
`\ifinstanzaR` 769, 770, 1560, 1576, 2285
`\ifl@d@dash` 1392
`\ifl@d@elin` 1394, 1395
`\ifl@d@esl` 1395
`\ifl@d@pnum` 1389, 1393
`\ifl@d@ssub` 1391
`\ifl@dpagefull` 2068, 2103, 2204
`\ifl@dpadding` 13, 1565, 1575
`\ifl@dpairing` 13, 77, 1403
`\ifl@dsamelang` 1687
`\ifl@dsamepage` 2044, 2078, 2204
`\ifl@dskipnumber` 1244
`\ifl@dusedbabel` 1685
`\iflabelpstart` 858, 901
`\ifledgroupnotesL@` 1093
`\ifledgroupnotesR@` 1075, 1243
`\iflednopbinverse` 2275, 2286
`\ifledplinenum` 1390
`\ifledRcol` 13, 179,
 201, 205, 209, 213, 257, 274,
 338, 347, 372, 386, 403, 420,
 432, 440, 461, 506, 533, 542,
 551, 618, 624, 630, 637, 645,
 650, 654, 659, 670, 696, 719,
 1407, 1446, 1513, 1526, 1713, 1725
`\ifluatex` 1527, 1538
`\ifnoteschanged@` 87
`\ifnumberedpar@`
 ... 831, 875, 911, 932, 1511, 1525
`\ifnumbering` 148, 827, 908
`\ifnumberingR` ... 43, 74, 112, 871, 929
`\ifnumberline` 1075, 1093, 1243
`\ifnumberpstart` ... 849, 893, 920, 941
`\ifnumequal` 1468
`\ifnumgreater` 1476, 1496, 1503
`\ifodd` 1277, 1302,
 1323, 1491, 2136, 2142, 2145, 2152
`\ifparledgroup` 2464, 2496
`\ifprint@last@after@pendL`
 ... 950, 2060, 2222
`\ifprint@last@after@pendR`
 ... 950, 2095, 2243
`\ifpst@rtedL` 38, 835
`\ifpst@rtedR` 38, 879
`\ifpstartnum` 1333, 1338
`\ifpstartnumR` 1288
`\ifshiftedpstarts` 5, 2053, 2088, 2296
`\ifsidepstartnum` 850, 894, 1290, 1311
`\ifstrempty` 860, 903, 924, 945
`\IfStrEq` 974, 1045, 1879,
 1887, 2251, 2255, 2263, 2267,

- 2278, 2279, 2289, 2290, 2414,
2415, 2422, 2423, 2435, 2436, 2443
`\ifstrequal` 2467, 2473
`\ifsublines@` 233,
326, 371, 404, 411, 442, 451,
463, 470, 482, 516, 572, 574,
1076, 1094, 1160, 1247, 1448, 1452
`\ifvbox` 967, 1038, 1748, 1751, 2309, 2355
`\ifvmode` 1414, 1426
`\ifwidthliketwocolumns` 11
`\ifwrittenlinesL` 2305, 2317
`\ifwrittenlinesR` 2306, 2363
`\initnumbering@sectcmd` 742, 756
`\initnumbering@sectcountR`
..... 66, 99, 119, 2079
`\InputIfFileExists` 69
`\insert@count` 548, 671, 697, 1520, 1547
`\insert@countR` 549, 670, 696, 1516, 1536
`\insert@noterule@ledgroup`
..... 2418, 2426, 2434
`\inserthangingsymbolfalse` 980
`\inserthangingsymbolL` ... 1002, 1551
`\inserthangingsymbolR` 1551
`\inserthangingsymbolRfalse` 1051
`\inserthangingsymbolRtrue` 1049
`\inserthangingsymboltrue` 978
`\insertlines@listR`
.... 80, 243, 259, 554, 1359, 1363
`\inserts@list` 837, 1519, 1546
`\inserts@listR` 881, 1354,
1357, 1367, 1381, 1382, 1515, 1535
`\instanzaLfalse` 1854, 2127
`\instanzaLtrue` 780
`\instanzaRfalse` 1855, 2128
`\instanzaRtrue` 803
`\interlinepenalty` 1594
`\itemcount@` 1466, 1468,
1473, 1476, 1494, 1496, 1501, 1503
- L**
- `\ld@nums` 722, 725, 731, 734
`\ld@set` 419, 645, 646
`\ld@dbbl@set@language` 1687, 1710
`\ld@dbfnote` 1510
`\ldc@maxchunks` 843, 845,
887, 889, 1644, 1654, 1662, 1669
`\ld@dcalc@maxoftwo`
..... 2020, 2191, 2338, 2380
`\ld@dcalc@minoftwo` .. 2028, 2115, 2191
`\ld@dcalcnum` 1181
`\ld@checklang` 1687
`\ld@dchset@num` 293, 296, 419
`\ld@dcnotetext` 1475, 1478
`\ld@dcnotetext@l` 1478, 1495
`\ld@dcnotetext@r` 1478, 1502
`\ld@emptyd@ta` 971, 1042
`\ld@dend@stuff` 63, 117, 133, 141
`\ld@getline@margin` 177
`\ld@getsidenote@margin` 1457
`\ld@dld@ta` 999,
1268, 1280, 1293, 1305, 1314, 1326
`\ld@leftbox`
.. 952, 982, 992, 1816, 2054, 2057
`\ld@linenumR` 225
`\ld@dl@sn@te` 1001
`\ld@make@labels` 1430
`\ld@make@labelsR` 1418, 1435
`\ld@minpagelines`
.... 1989, 2029, 2116, 2213, 2234
`\ld@dumpstartsL` 842,
843, 845, 847, 861, 862, 925,
926, 1648, 1680, 1735, 1774,
1775, 1851, 2000, 2001, 2125, 2321
`\ld@dumpstartsR` 47, 886,
887, 889, 891, 904, 905, 946,
947, 1648, 1681, 1738, 1774,
1775, 1852, 2000, 2001, 2126, 2367
`\ld@doldbbl@set@language` 1709
`\ld@doldselectlanguage` 1708, 1712, 1717
`\ld@dpagfullfalse`
.... 2204, 2253, 2258, 2265, 2270
`\ld@dpagfulltrue`
.... 2204, 2252, 2257, 2264, 2269
`\ld@dpagingfalse` 14, 741, 761
`\ld@dpagingtrue` 755
`\ld@dpairingfalse` 13, 745, 760
`\ld@dpairingtrue` 740, 754
`\ld@dpscL` 967, 972,
985, 1009, 1012, 1014, 1650,
1682, 1735, 1748, 1783, 1789,
1795, 1798, 1800, 1849, 2010,
2016, 2021, 2024, 2032, 2046,
2047, 2049, 2061, 2062, 2123,
2309, 2311, 2314, 2321, 2332,
2333, 2338, 2340, 2342, 2413, 2468
`\ld@dpscR` 1038, 1043, 1056,
1651, 1683, 1738, 1751, 1784,
1790, 1806, 1809, 1811, 1850,
2011, 2017, 2025, 2033, 2081,
2082, 2084, 2096, 2097, 2124,

- 2355, 2357, 2360, 2367, 2376,
2377, 2380, 2382, 2384, 2421, 2470
\l@drd@ta 1002,
1270, 1278, 1295, 1303, 1316, 1324
\l@drightbox
952, 1053, 1064, 1821, 2089, 2092
\l@drsn@te 1003
\l@dsamepagefalse
.... 2204, 2252, 2257, 2264, 2269
\l@dsamepagetrue
.... 2204, 2253, 2258, 2265, 2270
\l@dsetupmaxlinecounts .. 1661, 1678
\l@dsetuprawboxes 1653, 1677
\l@dskipnumberfalse 1245
\l@dskipnumbertrue 1141
\l@dunhbox@line 1002, 1017, 1020
\l@dusedbabelfalse 1685, 1705
\l@dusedbabeltrue 1685, 1707
\l@duselanguage
.... 1697, 1793, 1804, 2042, 2077
\l@dzeromaxlinecounts ... 1661, 1679
\l@dzeropenalties 914, 935, 1778, 2005
\l@prev@nopbR ... 54, 2400, 2407, 2408
\l@prev@pbR 53, 2399, 2405, 2406
\l@pscL 1650
\l@pscR 1650
\label@refs
1410, 1412, 1418, 1422, 1424, 1430
\labelref@list 1421, 1424, 1453
\labelref@listR 1400, 1409, 1412, 1449
\language name .. 1688, 1689, 1691–1694
\last@page@num 358, 364
\last@page@numR 344
\lastbox 975, 1046
\lastskip 616, 622
\lcolwidth
.. 6, 7, 16, 757, 856, 983, 992,
1801, 1916, 1937, 1954, 1965, 1981
\led@err@BadLeftRightPstarts ...
..... 29, 1775, 2001
\led@err@LeftOnRightPage ... 32, 2146
\led@err@LineationInNumbered ... 149
\led@err@ManyLeftnotes 1496
\led@err@ManyRightnotes 1503
\led@err@ManySidenotes 1476
\led@err@NumberingNotStarted ... 91
\led@err@numberingShouldHaveStarted
..... 121
\led@err@NumberingStarted 44
\led@err@PendNoPstart 912, 933
\led@err@PendNotNumbered ... 909, 930
\led@err@PstartInPstart ... 832, 876
\led@err@PstartNotNumbered . 828, 872
\led@err@RightOnLeftPage ... 32, 2153
\led@err@TooManyPstarts
..... 24, 26, 844, 888
\led@mess@NotesChanged 88
\led@mess@SectionContinued
..... 115, 131, 139
\led@nopbnumR 2404, 2405
\led@nopbR 2403, 2405
\led@pb@setting
..... 1879, 1887, 2251, 2255,
2263, 2267, 2278, 2279, 2289, 2290
\led@pbnumR 2402, 2405
\led@pbR 2401, 2405
\led@warn@BadAction 1143
\led@warn@BadAdvancelineLine 389, 395
\led@warn@BadAdvancelineSubline .
..... 375, 381
\led@warn@BadLineation 166
\led@warn@BadSetline 635
\led@warn@BadSetlinenum 643
\led@warn@DuplicateLabel 1437
\ledgroupnotesL@false 2429
\ledgroupnotesL@true 2417
\ledgroupnotesR@false 2432
\ledgroupnotesR@true 2425
\ledllfill 1002
\lednopb 799
\lednopbnum 2278, 2401
\lednopbnumR 2289, 2401
\lednopbR 799, 2403
\ledpb 798
\ledpbnum 2279
\ledpbnumR 2290, 2401
\ledpbR 798, 2401
\ledRcol@false 1067
\ledRcol@true 1036
\ledRcolfalse 15, 772, 805
\ledRcoltrue 789
\ledrlfill 1002, 1925, 1947
\ledsavedprintlines 10, 1387
\ledsectnomark 1008
\ledsectnotoc 1007, 2045, 2080
\ledstrutL 2054, 2057, 2131
\ledstrutR 2089, 2092, 2131
\ledthegoal 2212, 2233, 2295
\leftlinenumR 225, 1268, 1280
\leftpstartnumL 1288

`\leftstartnumR` 1288
`Leftside` (environment) 7, 771
`\Leftsidehook` 778, 783
`\Leftsidehookend` 782, 783
`\line@list` 729, 733
`\line@list@stuff` 132
`\line@list@stuffR` .. 62, 116, 140, 598
`\line@listR` . 83, 243, 258, 574, 720, 724
`\line@margin` 182, 1298
`\line@marginR` 175, 1273, 1319
`\line@num` . 361, 393, 394, 396, 414,
425, 426, 454, 581, 1100, 1451, 2326
`\line@numR` 55,
232, 239, 298, 332, 351, 387,
388, 390, 407, 421, 422, 445,
567, 571, 585, 1082, 1154, 1163,
1252, 1254, 1256, 1257, 1447, 2372
`\lineation` 171, 172, 800
`\lineation*` 8, 171
`\lineationR` 8, 147, 173, 800
`\linenum@out`
. 613, 619, 625, 631, 638, 646,
651, 655, 1420, 1759, 1836, 2185
`\linenum@outR` 595, 601,
603, 608, 609, 615, 618, 624,
630, 637, 645, 650, 654, 659,
1408, 1764, 1840, 2188, 2401–2404
`\linenumberlist` 1253, 1257
`\linenumincrement` 7, 196
`\linenumincrement*` 7, 196
`\linenummargin` 175
`\linenumr@p` 1390, 1394, 1447, 1451
`\linenumrepR` 222, 232
`\linenumsep`
. 227, 229, 1335, 1338, 1347, 1350
`\linesinpar@listL`
..... 250, 268, 582, 2159, 2162
`\linesinpar@listR`
..... 250, 262, 586, 2165, 2168
`\linesonpage@listL` 269, 590, 2172, 2175
`\linesonpage@listR` 263, 593, 2178, 2181
`\list@clear`
. 258–265, 268, 269, 271, 837, 881
`\list@clearing@reg` 267
`\list@create`
... 243–248, 250–252, 1354, 1400
`\listxadd` 366, 2405–2408
`\lock@disp` 1213, 1217, 1222
`\lock@off` 503, 504, 512, 654, 655
`\lock@on` 650, 651

M

`\managestanza@modulo` 1620
`\maxchunks` 5, 1644
`\maxlinesinpar@list` 250, 271
`\memorydump` 9, 777, 794
`\memorydumpL` 126, 777
`\memorydumpR` 126, 794
`\message` 61
`\multiply` 1187, 1919, 1968

N

`\n@num` 540, 659
`\n@num@reg` 546
`\namebox` 967, 972, 1038,
1043, 1628, 1748, 1751, 2309, 2355
`\NeedsTeXFormat` 2
`\new@line` 1017, 1020
`\new@lineL` 612, 1002
`\new@lineR` 614
`\newbox` 810, 952, 953, 1629
`\newcommandx` 820, 864, 907, 928
`\newcounter` 99–102, 187,
189, 191, 193, 813, 814, 816, 817
`\newif` 5, 39, 143,
144, 596, 769, 770, 950, 951,
1342, 1551, 1685, 1732, 1744,
1877, 1878, 2204, 2206, 2305, 2306
`\newlength` 1905, 1908
`\newmarks` 2409–2411
`\newnamebox` 1628, 1656, 1657
`\newnamecount` 1639, 1664
`\newsavebox` 1768, 1769
`\newwrite` 595, 2398
`\next@absline`
.... 1880, 1882, 1884, 2256–2258
`\next@abslineR`
.... 1881, 1883, 1885, 2268–2270
`\next@action` 288
`\next@actionline` 285, 287
`\next@actionlineR`
. 277, 279, 1112, 1150, 1172, 1174
`\next@actionR` 280, 1113,
1151, 1152, 1157, 1158, 1166, 1175
`\next@insert` 838
`\next@insertR`
882, 1358, 1361, 1363, 1366, 1370
`\next@page@num` 365, 437
`\next@page@numR` 59, 301, 303, 355, 434
`\no@expands` 668, 694
`\noindent` 862, 905, 926, 947

- \normal@page@break 366
 - \normal@page@breakR 52
 - \normal@pars 76, 841, 885
 - \normalbfnoteX 1524
 - \notefontsetup 2455
 - \noteschanged@true
 - 81, 84, 721, 730, 1360
 - \notesXwidthliketwocolumns 6
 - \num@lines 915, 1779, 2006
 - \num@linesR 809, 936, 1780, 2007
 - \numberedpar@true 857, 900
 - \numberingRfalse 75
 - \numberingRtrue 49, 110, 136
 - \numberingtrue 128
 - \numberpstartfalse 10
 - \numberpstarttrue 10
 - \numdef 1009, 1494, 1501,
 - 1880, 1881, 2256, 2268, 2468, 2470
 - \numgdef 1466, 1473, 2277, 2288
 - \numlabfont 232
 - \numpagelinesL 1989,
 - 2052, 2069, 2073, 2213, 2279, 2488
 - \numpagelinesR
 - 1989, 2087, 2104, 2108, 2234, 2290
- O**
- \old@otherlanguage 1722
 - \old@startstanza .. 779, 780, 802, 803
 - \oldchapter 752, 762
 - \one@line .. 972, 975, 1002, 1017, 1020
 - \one@lineR 809, 1043, 1046
 - \openout 70, 603, 609
 - \otherlanguage 1722, 1723
- P**
- \p@pstartL 859
 - \p@pstartR 902
 - \PackageError 21
 - \page@action 302, 431, 560
 - \page@num 283, 363, 1300
 - \page@numR 254, 275, 353,
 - 566, 571, 1152, 1275, 1321, 1489
 - \pagebreak 1894
 - \pagegoal 2303
 - \Pages 7, 1993
 - pages (environment) 6, 739
 - \pagetotal 2138, 2212, 2233
 - pairs (environment) 5, 739
 - \paperwidth 1016, 1019
 - \par@line 916, 1781, 2008
 - \par@lineR 809, 937, 1782, 2009
 - \parbox 1801, 1812
 - \parledgroup@ .. 974, 1045, 2409, 2435
 - \parledgroup@beforenotes@save ..
 - 923, 944, 2495
 - \parledgroup@beforenotesL 2493
 - \parledgroup@beforenotesR 2493
 - \parledgroup@correction@notespacing
 - 2056, 2091, 2482
 - \parledgroup@correction@notespacing@final
 - 2344, 2386, 2463
 - \parledgroup@correction@notespacing@init
 - 2074, 2109, 2458, 2466
 - \parledgroup@notes@endL
 - 2312, 2315, 2343, 2428
 - \parledgroup@notes@endR
 - 2358, 2361, 2385, 2431
 - \parledgroup@notes@startL
 - 974, 2412, 2428
 - \parledgroup@notes@startR
 - 1045, 2412, 2428
 - \parledgroup@notespacing@correction
 - 2453, 2485–2487
 - \parledgroup@notespacing@correction@accumulated
 - 2459, 2465, 2486
 - \parledgroup@notespacing@correction@modulo
 - 2460, 2487–2489
 - \parledgroup@notespacing@set@correction
 - 1997, 2453
 - \parledgroup@series
 - 2410, 2414, 2415,
 - 2422, 2423, 2438, 2439, 2445, 2446
 - \parledgroup@type 2411,
 - 2414, 2415, 2422, 2423, 2436, 2443
 - \parledgroupnotespacing . 2452, 2455
 - \parledgroupseries@ 2409
 - \parledgrouptrue 10
 - \parledgrouptype@ 2409
 - \pausenumbering 792
 - \pausenumberingR 109, 792
 - \pend 8, 776, 797, 833, 1607
 - \pendL 776, 907
 - \pendR 797, 877, 928
 - \prev@abslineverse
 - 2277–2279, 2288–2290
 - \prev@nopbR 2399
 - \prev@pbR 2399
 - \prevgraf
 - 915, 936, 1779, 1780, 2006, 2007
 - \print@columnseparator .. 1820, 1857

- `\print@eledsectionL` 997, 1005, 1801, 2050
`\print@eledsectionR` . 1071, 1812, 2085
`\print@last@after@pendLfalse` .. 2225
`\print@last@after@pendLtrue` ... 2348
`\print@last@after@pendRfalse` .. 2246
`\print@last@after@pendRtrue` ... 2388
`\print@lineL` 987, 997
`\print@lineR` 1058, 1071
`\printlines` 1398
`\printlinesR` 10, 1387
`\ProcessOptions` 12
`\protected@edef` 858, 901
`\protected@write` ... 1417, 1429, 1691
`\protected@xdef` 1533, 1544
`\ProvidesPackage` 3
`\pst@rtedLfalse` 38
`\pst@rtedLtrue` 129, 839
`\pst@rtedRfalse` 40, 48, 78
`\pst@rtedRtrue` 113, 137, 883
`\pstart` 8, 27, 31, 774, 796, 1609
`\pstartL` 774, 812
`\pstartnumfalse` 1335, 1340
`\pstartnumRfalse` 1347, 1352
`\pstartnumRtrue` 1343, 1788, 2379
`\pstartnumtrue` 1787, 2337
`\pstartR` 796, 812
- R**
- `\Rcolwidth` 6, 7, 16, 758, 899, 1054, 1064, 1812, 1917, 1938, 1955, 1966, 1982
`\read@linelist` 256, 599
`\rem@inder` 1257, 1259–1261
`\resetprevline@` 1837, 1841, 2327, 2373
`\resumenumbering` 793
`\resumenumberingR` 109, 793
`\rightlinenumR` 225, 1270, 1278
`\rightpstartnumL` 1288
`\rightpstartnumR` 1288
`Rightside` (environment) 7, 788
`\Rightsidehook` 783, 801
`\Rightsidehookend` 783, 806
`\rlap` 1270, 1278, 1295, 1303, 1316, 1324
`\Rlineflag` 9, 220, 232, 1390, 1394, 1439
`\rule` 1897
- S**
- `\savebox` 1801, 1812
`\sc@n@list` 1258, 1260
`\secdef` 767
`\section@num` 130–132
`\section@numR` 36, 50, 61, 62, 69, 70, 114–116, 138–140
`\select@language` 1689, 1691–1694, 1728
`\selectlanguage` 1697
`\set@line` 669, 695, 718
`\set@line@action` 295, 400, 409, 416, 439, 562
`\setl@dlp@rbox` 1482, 1497, 1499
`\setl@drp@rbox` 1484, 1492, 1504
`\setline` 633
`\setlinenum` 641
`\setnamebox` 847, 891, 1628
`\setnotepositionliketwocolumns@C` 1911
`\setnotepositionliketwocolumns@L` 1911
`\setnotepositionliketwocolumns@R` 1911
`\setnotespositionliketwocolumns@C` 1950
`\setnotespositionliketwocolumns@L` 1928
`\setnotespositionliketwocolumns@R` 1977
`\setpositionliketwocolumns@C` ... 1911, 1945
`\setpositionliketwocolumns@L` ... 1911, 1924
`\setpositionliketwocolumns@R` ... 1911, 1973
`\setprintlines` 1388
`\setwidthliketwocolumns@C` 1911, 1932
`\setwidthliketwocolumns@L` 1911, 1911
`\setwidthliketwocolumns@R` 1911, 1962
`\shiftedpstartsfalse` 7
`\shiftedpstartstrue` 6, 8, 9
`\shiftedversesfalse` 7
`\shiftedversestrue` 6
`\showlemma` 678, 704
`\sidenote@margin` 1459, 1462
`\sidenote@marginR` 1456, 1487
`\sidenotecontent@` 1465, 1470, 1471, 1482, 1484, 1492, 1493, 1497, 1499, 1500, 1504
`\sidenotemargin` 1456
`\sidenotemargin*` 1456
`\sidenotesep` 1471
`\skip` 2438, 2445

- `\skip@lockoff` 504, 512
`\skipnumbering` 10, 658
`\skipnumbering@reg` 662
`\smash` 1897
`\splitbotmarks` 2435,
2436, 2438, 2439, 2443, 2445, 2446
`\splitfirstmarks`
974, 1045, 2414, 2415, 2422, 2423
`\splittopskip` 969, 1040
`\stanza@count` 1589, 1603, 1615
`\stanza@hang` 1591, 1623
`\stanza@modulo` 1589, 1618
`\stanzaindentbase`
..... 1569, 1579, 1616, 1619
`\startlock` 649
`\startstanzahook` 1587
`\startsub` 616
`\sub@action` 311, 460, 561
`\sub@change` 60, 305, 306, 312
`\sub@lock` 1095
`\sub@lockR` 57, 320, 322, 324,
327, 478, 484, 485, 487, 488,
518, 519, 521, 1077, 1133, 1135,
1136, 1138, 1194, 1234, 1236, 1238
`\sub@off` 624, 625
`\sub@on` 618, 619
`\subline@num` 234, 361, 379,
380, 382, 412, 452, 1096, 1101, 1452
`\subline@numR` 235,
239, 328, 332, 351, 373, 374,
376, 405, 443, 568, 572, 1078,
1083, 1154, 1161, 1248, 1249, 1448
`\sublinenumincrement` 7, 196
`\sublinenumincrement*` 7, 196
`\sublinenumr@p` . 1391, 1395, 1448, 1452
`\sublinenumrepR` 222, 235
`\sublines@false` 58, 309, 1123
`\sublines@true` 307, 1121
`\sublock@disp` 1196, 1200, 1205
`\sw@list@inedtextR` 248, 265
`\sw@listR` 247, 264
`\symplinenum` 1390
`\sza@penalty` 1598, 1602
- T**
- `\temp` 1914, 1915, 1918–1921,
1934, 1935, 1940–1942, 1951,
1953, 1956–1959, 1963, 1964,
1967–1970, 1978, 1980, 1983–1986
`\temp@` 1009, 1010
- `\temp@spacing` 2455, 2456
`\tempa` 1952, 1954–1956, 1979, 1981–1983
`\textwidth` 17, 19, 757, 758
`\theledlanguageL` ... 1697, 1793, 2042
`\theledlanguageR` ... 1697, 1804, 2077
`\thepage` 613, 615, 1418, 1430
`\thepstart` 775, 795
`\thepstartL`
10, 775, 815, 852, 859, 1334, 1339
`\thepstartR`
10, 795, 818, 895, 902, 1346, 1351
`\thisfootnote` .. 1533, 1534, 1544, 1545
`\thr@@` .. 487, 496, 519, 526, 1128, 1136
`\topskip` 2138
- U**
- `\unhbox` 1633,
1816, 1821, 2054, 2057, 2089, 2092
`\unhnamebox` 1628
`\unvbox` 975, 1046, 1635
`\unvnamebox` 1628
`\usebox` 1818, 1823
`\usernamecount` 1590, 1597, 1639, 1671,
2021, 2311, 2314, 2338, 2340,
2357, 2360, 2380, 2382, 2413, 2421
- V**
- `\value` 836, 880, 1614,
1772, 1773, 1995, 1996, 2324, 2370
`\vbadness` 968, 1039
`\vbfnoteX` 1534, 1545
`\vbox` 847, 891, 1801, 1812
`\vl@dbfnote` 1514, 1518
`\vsplit` 972, 1043
- W**
- `\wd` 1002
`\widthliketwocolumns` 6
`\widthliketwocolumnstrue` 11
`\WithSuffix` 171, 216–219, 1456
`\writtenlinesLfalse` 2012, 2322
`\writtenlinesLtrue` 2319
`\writtenlinesRfalse` 2013, 2368
`\writtenlinesRtrue` 2365
- X**
- `\x@lemma` 680–682, 706–708
`\xifinlist` 985,
1010, 1056, 1795, 1806, 2049, 2084

<code>\xifinlistcs</code>	1882–	441, 448, 450, 457, 462, 464,
	1885, 1888–1891, 2252, 2253,	466, 469, 471, 473, 481, 483,
	2257, 2258, 2264, 2265, 2269, 2270	492, 515, 517, 524, 543, 544,
<code>\Xnoteswidthliketwocolumns</code>	6	554, 570, 582, 586, 590, 593,
<code>\xpg@main@language</code>	1729, 1730	1447, 1451, 1514, 1518, 1534, 1545
<code>\xright@appenditem</code>		Z
	433, 434, 436, 437,	<code>\zz@@@</code>
		1410, 1422

Change History

v0.1		v0.12
General: First public release	1	General: New new management of hangingsymbol insertion, pre- venting undesirable insertions. 58
v0.10		v0.2
General: <code>\edlabel</code> commands on the right side are now correctly indicated.	1	General: Added section of babel re- lated code
<code>\edlabel</code> commands which start a paragraph are now put in the right place.	1	Fix babel problems
v0.11		<code>\Columns</code> : Added <code>\l@dchecklang</code> and <code>\l@duselanguage</code> to <code>\Columns</code>
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).	43	<code>\Pages</code> : Added <code>\l@duselanguage</code> to <code>\Pages</code>
Lineation can be by <code>pstart</code> (like in <code>eledmac</code> 0.15).	19	v0.3
New management of hang- ingsymbol insertion, preventing undesirable insertions.	58	General: Added <code>\do@lineLhook</code> and <code>\do@lineRhook</code>
Prevent shift of column separator when a verse is hanged	58	Reorganize for <code>ledarab</code>
<code>\affixline@numR</code> : Changed <code>\affixline@numR</code> to allow to disable line numbering (like in <code>eledmac</code> 0.15).	48	<code>\affixline@numR</code> : Changed <code>\affixline@numR</code> to match new <code>eledmac</code>
<code>\Columns</code> : Line numbering by <code>pstart</code>	67	<code>\do@actions@nextR</code> : Used <code>\do@actions@fixedcode</code> in <code>\do@actionsR</code>
<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).	80	<code>\do@lineL</code> : Added <code>\do@lineLhook</code> to <code>\do@lineL</code>
<code>Pstart</code> number can be printed in side	80	Simplified <code>\do@lineL</code> by using macros for some common code 43
		<code>\do@lineR</code> : Changed <code>\do@lineR</code> similarly to <code>\do@lineL</code>
		<code>Leftside</code> : Added hooks into Left- side environment
		<code>\flag@end</code> : Removed extraneous spaces from <code>\flag@end</code>
		37
		32

<code>\ifledRcol</code> : Moved <code>\ifl@dpairing</code> to <code>eledmac</code>	15	v0.5	General: Corrections about <code>\section</code> and other titles in numbered sections	1
<code>\ifpst@rtedR</code> : Moved <code>\ifpst@rtedL</code> to <code>eledmac</code>	16			
<code>\l@dlinenumR</code> : Simplified <code>\leftlinenumR</code> and <code>\rightlinenumR</code> by introducing <code>\l@dlinenumR</code>	22	v0.6	General: Be able to use <code>\chapter</code> in parallel pages.	1
<code>\l@dnumpstartsR</code> : Moved <code>\l@dnumpstartsL</code> to <code>eledmac</code>	61	v0.7	General: Option ‘ <code>shiftedverses</code> ’ which make there is no blank between two parallel verses with unequal length.	1
<code>\ledsavedprintlines</code> : Simplified <code>\printlinesR</code> by using <code>\setprintlines</code>	54	v0.8	General: Possibility to have a symbol on each hanging of verses, like in the french typography. Redefine the commande <code>\hangingsymbol</code> to define the character.	1
<code>\ledstrutR</code> : Added <code>\ledstrutL</code> and <code>\ledstrutR</code>	75			
<code>\normalbfnoteX</code> : Removed extraneous spaces from <code>\normalbfnoteX</code>	57			
<code>\Pages</code> : Added <code>\ledstrutL</code> to <code>\Pages</code>	73			
Added <code>\ledstrutR</code> to <code>\Pages</code>	74	v0.9	General: Possibility to number <code>\pstart</code>	10
<code>\Rightsidehookend</code> : Added <code>\Leftsidehook</code> , <code>\Leftsidehookend</code> , <code>\Rightsidehook</code> and <code>\Rightsidehookend</code>	37		Possibility to number the <code>pstart</code> with the commands <code>\numberpstarttrue</code>	1
<code>\sublinenumrepR</code> : Added <code>\linenumrepR</code> and <code>\sublinenumrepR</code>	21		<code>\ifledRcol</code> : Moved <code>\iflledRcol</code> and <code>\ifnumberingR</code> to <code>eledmac</code>	15
v0.3.a		v0.9.1	General: The numbering of the <code>pstarts</code> restarts on each <code>\beginnumbering</code>	1
General: Minor <code>\linenummargin</code> fix	1			
v0.3.b		v0.9.2	General: Debug : with <code>\Columns</code> , the hanging indentation now runs on the left columns and the hanging symbol is shown only when <code>\stanza</code> is used.	1
General: Improved parallel page balancing	1			
v0.3.c		v0.9.3	General: <code>\thepstartL</code> and <code>\thepstartR</code> use now <code>\bfseries</code> and not <code>\bf</code> , which is deprecated and makes conflicts with memoir class.	1
General: Compatibility with Polyglossia	1			
v0.3a				
<code>\line@marginR</code> : Don’t just set <code>\line@marginR</code> in <code>\linenummargin</code>	20			
v0.3b		v1.0	General: Compatibility with <code>eledmac</code> . Change name to <code>eledpar</code>	1
<code>\Pages</code> : Added <code>\l@dminpagelines</code> calculation for succeeding page pairs	74		Debug in lineation by <code>pstart</code>	19
v0.4		v1.0.1	General: Correction on <code>\numberonlyfirstinline</code> with lineation by <code>pstart</code> or by	
General: No more <code>ledparpatch</code> . All patches are now in the main file.	1			

page.	1	v1.3.3	General: Debugging the left notes of the right column.	56
v1.1			\l@dbfnote : Spurious space with footnote in right column. . . .	57
General: Shiftedverses becomes shiftedpstarts.	1	v1.3.4	General: Allow use of commands in sidenotes, as introduced by eledmac 1.0.	56
\pstartR : Add \labelpstarttrue (from eledmac).	39	v1.3.5	\normalbfnoteX : Allows one to redefine \thefootnoteX with alph when some packages are loaded.	57
v1.1.1		v1.4	General: Added \do@insidelineLhook and \do@insidelineRhook . . .	44
\pstartR : Correct \pstartR bug introduced by 1.1.	39	v1.4.1	\normalbfnoteX : Fix bug with normal familiar footnotes when mixing RTL and LTR text. . .	57
v1.1.2		astanza : Enable the use of stanzaidentsrepetition within stanza environment.	59	
\affixside@noteR : Remove spurious space between line number and line content	56	v1.4.3	General: Corrects a false hanging verse when a verse is exactly the length of a line.	1
v1.10.0		\inserthangingsymbolR : Hang verse is now not automatically flush right.	58	
General: Compatibility with \AtEveryPstart and \AtEveryPend	1	\pendL : Spurious spaces in \pendL	41	
Restore critical notes in \eledsection in parallel columns (this bug was added in 1.8.2).	1	\pendR : Spurious spaces in \pstartR	42	
\edlabel : Prevent some bugs with cross-referencing when using \autopar in parallel texts. . .	54	\pstartR : Spurious spaces in \pstartL and \pstartR	39	
\Pages : Debug wrong pages splitting when no optional argument is used in last \pend (bug was added in v.1.8.3).	71	v1.5.0	General: Add, as in eledmac, features to manage page breaks. . .	1
Debug wrong parallel pages synchronization when an \edtext falls accross two pages.	71	\sublinenumincrement* : Add starred version of \firstlinenum , \linenumincrement , \firstsublinenum , \sublinenumincrement to change both Left and Right-side.	21	
v1.10.1		v1.6.0	General: Add tool and documentation for parallel ledgroups . . .	12
\line@list@stuffR : Revert modification of 1.4.2 which makes bug with numbering. Leave vertical mode to solve spurious space before minipage.	32			
v1.2				
General: Support for \led<section> commands in parallel texts. . .	1			
v1.2.1				
\initnumbering@sectcountR : For the right section, the counter is defined only once.	18			
v1.3				
General: Manage RTL language. .	34			
v1.3.2				
General: Debug with some classes. .	1			

v1.7.0	
General: Add, as in eledmac, features to make crossrefs with pstart numbers.	1
v1.8.0	
General: <code>\beginnumbering</code> is defined only on eledmac, not on eledpar.	16
<code>\l@dlsnote</code> , <code>\l@drsnote</code> and <code>\l@dcsnote</code> defined only one time, in eledmac.	56
Add <code>\beforecolumnseparator</code> and <code>\aftercolumnseparator</code> . . .	6
Add <code>\columnspan</code>	6
Add, as in eledmac, new system of sectioning commands.	1
Add, as in eledmac, option to insert something after <code>\pends</code> / verses.	1
Add, as in eledmac, option to insert something between <code>\pstarts</code> / verse.	1
Change <code>\do@lineR</code> and <code>\do@lineL</code> to allow new sectioning commands.	43
Compatibility with musixtex. . .	1
Debug eledmac sectioning command after using <code>\resumenumbering</code>	1
New sectioning commands, as in eledmac.	14
<code>\Columns</code> : Modify <code>\Columns</code> to enable to add section's title. . .	65
Suppress <code>\l@dchecklang</code> from <code>\Columns</code>	66
<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.	62
<code>\Pages</code> : Modify <code>\Pages</code> to enable to add section's title.	71
<code>\pendL</code> : As in eledmac, <code>\pendL</code> can have an optional argument. . .	41
<code>\pendR</code> : As in eledmac, <code>\pendR</code> can have an optional argument. . .	42
<code>\print@columnseparator</code> : Move some code of <code>\Columns</code> to <code>\print@columnseparator</code> . . .	67
<code>\pstartR</code> : As in eledmac, <code>\pendL</code> and <code>\pendR</code> can have an optional argument.	39
<code>\sidenotemargin*</code> : <code>\sidenotemargin</code> is now directly defined in eledmac to be able to manage eledpar.	56
Add <code>\sidenotemargin*</code>	56
<code>\theledlanguageR</code> : Correct left/right language setting with polyglossia.	64
v1.8.1	
<code>\do@lineL</code> : Fix a bug with critical notes a the begining of a page, (maybe added by v1.8.0) (?). .	43
<code>\do@lineR</code> : Fix a bug with critical notes a the begining of a page, added by v1.8.0 (?).	44
v1.8.2	
General: Debug <code>\eledxxx</code> with some paper sizes	1
Debug left and side note (bugs added by 1.8.0)	1
<code>\eledpar@error</code> : Errors specific to eledpar send to eledpar handbook	16
<code>\flag@end</code> : <code>\flag@start</code> and <code>\flag@end</code> are now defined only one time for eledmac and eledpar	32
<code>\lineation*</code> : Add <code>\lineation*</code> .	20
v1.8.3	
General: Add <code>\noeledxxx</code> , as in eledmac	1
<code>\doinsidelineRhook</code> : Added <code>\dolineLhook</code> , <code>\dolineRhook</code> , <code>\doinsidelineLhook</code> and <code>\doinsidelineRhook</code>	44
<code>\Pages</code> : Debug blank pages when using optional argument in the last <code>\pend</code>	71
<code>\resumenumberingR</code> : Debug <code>\resumenumberingR</code>	18
v1.9.0	
General: Add <code>\AtBeginPairs</code> macro.	5
Compatibility with <code>\Xnoteswidthliketwocolumns</code> and <code>\notesXwidthliketwocolumns</code>	1

<code>\ifwidthliketwocolumns:</code>	Added		.aux file when setting left/right	
	widthliketwocolumns option . .	15	lines.	64
<code>\theledlanguageR:</code>	Debug	v1.9.1		
	left/right language switching		<code>\ifledRcol:</code> Moved <code>\ifl@dpaging</code>	
	with polyglossia. Don't write in		to eledmac	15