

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 v1.8.3, last revised 2014/08/31.

[†]herries dot press at earthlink dot net

[‡]maieul at maieul dot net

5 Left and right texts	7
6 Numbering text lines and paragraphs	8
7 Verse	10
8 Side notes	12
9 Parallel ledgroups	12
9.1 Parallel ledgroups and <code>setspace</code> package	13
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	22
14.4 Commands within the line-list file	24
14.5 Writing to the line-list file	31
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	44
17.4 Line number printing	47
17.5 Pstart number printing in side	49
17.6 Add insertions to the vertical list	51
17.7 Penalties	51
17.8 Printing leftover notes	52
18 Footnotes	53
18.1 Normal footnote formatting	53
19 Cross referencing	53
20 Side notes	55

<i>List of Figures</i>	3
21 Familiar footnotes	56
22 Verse	57
23 Naming macros	59
24 Counts and boxes for parallel texts	60
25 Fixing babel	61
26 Parallel columns	63
27 Parallel pages	68
28 Sections' titles' commands	78
29 Page break/no page break, depending on the specific line	79
30 Parallel ledgroup	79
31 The End	82
Appendix A Some things to do when changing version	83
Appendix A.1 Migration to eledpar 1.4.3	83
References	83
Index	83
Change History	93

List of Figures

1 Introduction

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

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

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

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

2 The `eledpar` package

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

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

2.1 General

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

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

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

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

```
\maxchunks{5120}
```

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

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

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

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

3 Parallel columns

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

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

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

There is no required pagebreak before or after the columns.

`\Lcolwidth` The lengths `\Lcolwidth` and `\Rcolwidth` are the widths of the left and right columns, respectively. By default, these are:

```
\setlength{\Lcolwidth}{0.45\textwidth}
\setlength{\Rcolwidth}{0.45\textwidth}
```

They may be adjusted if one text tends to be 'bulkier' than the other.

<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.
<code>\columnspan</code>	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> <code>\aftercolumnseparator</code>	By default, the spaces around column separator are the same as the space: <ul style="list-style-type: none"> • On the left of columns, if columns are aligned right. • On the right of columns, if columns are aligned left. • On both the Left and Right columns, if columns are centered.

You can redefine `\beforecolumnseparator` and `\aftercolumnseparator` length to define spaces before or after the column separator, instead of letting `eledpar` calculate them automatically.

```
\setlength{\beforecolumnseparator}{length}
\setlength{\aftercolumnseparator}{length}
```

If you want to come back to the previous behavior, just set them with a negative value.

4 Facing pages

<code>pages</code>	Numbered text that is to be set on facing pages must be within a <code>pages</code> environment. Within the environment the text for the lefthand and righthand pages is placed within the <code>Leftside</code> and <code>Rightside</code> environments, respectively.
<code>\Pages</code>	The command <code>\Pages</code> typesets the texts in the previous pair of <code>Leftside</code> and <code>Rightside</code> environments. The general scheme for parallel pages looks like this:

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

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

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

\goalfraction When doing parallel pages **eledpar** has to guess where TeX is going to put pagebreaks and hopefully get there first in order to put the pair of texts on their proper pages. When it thinks that the fraction **\goalfraction** of a page has been filled, it finishes that page and starts on the other side's text. The definition is:
`\newcommand*{\goalfraction}{0.9}`

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

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

5 Left and right texts

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

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

Within these environments you can designate the line numbering scheme(s) to be used. The **eledmac** package originally used counters for specifying the numbering scheme; now both **eledmac**¹ and the **eledpar** package use macros instead. Following `\firstlinenum{<num>}` the first line number will be *<num>*, and following `\linenumincrement{<num>}` only every *<num>*th line will have a printed number. Using these macros inside the **Leftside** and **Rightside** environments gives you independent control over the left and right numbering schemes. The `\firstsublinenum` and `\sublinenumincrement` macros correspondingly set the numbering scheme for sublines. The starred versions change both left and right numbering schemes.

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

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

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

be specified within a `Leftside` or `Rightside` environment. A multi-chunk text then looks like:

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

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

Generally speaking, controls like `\firstlinenum` or `\linenummargin` apply to sequential and left texts. To effect right texts only they have to be within a `Rightside` environment. `\lineationR` macro is the equivalent of `eledmac \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 `eledpar` with the option `shiftedpstarts`.

6 Numbering text lines and paragraphs

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

```
\beginnumbering
<text>
```

`\endnumbering`

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

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

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

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

`\Rlineflag` The value of `\Rlineflag` is appended to the line numbers of the right texts. Its default definition is:

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

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

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

following code in your preamble:

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

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

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

7 Verse

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

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

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

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

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

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

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

```
\setstanzaindents{1,0,1,0,1,0,1,0,1,0,1}
```

```

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

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

  \end{astanza}
  ...

```

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

```

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

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

  \end{astanza}
  ...

```

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

```

\renewcommand{\hangingsymbol}{[,]}

```

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

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

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

8 Side notes

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

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

9 Parallel ledgroups

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

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

See the following example:

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

```

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

```

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

```

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

```

9.1 Parallel ledgroups and **setspace** package

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

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

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

10 Sectioning commands

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

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

11 Implementation overview

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

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

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

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

12 Preliminaries

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

```
1 {*code}
2 \NeedsTeXFormat{LaTeX2e}
3 \ProvidesPackage{eledpar}[2014/08/31 v1.8.3 eledmac extension for parallel texts]%
4
```

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

`\ifshiftedpstarts`

```

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

```

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

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

```

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

```

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

```

\l@dpairing
\l@dpaging
\ifledRcol
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 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@dnumpstartsR \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 `LATEX` 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         \initnumbering@sectcmd
119     \else
120         \led@err@numberingShouldHaveStarted
121         \endnumberingR
122         \beginnumberingR
123     \fi}
124

```

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

```

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

```

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.

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

```

146 \newcommand*{\lineationR}[1]{\if%
147   \ifnumbering
148     \led@err@LineationInNumbered
149   \else
150     \def\@tempa{#1}\def\@tempb{page}%
151     \ifx\@tempa\@tempb
152       \global\bypage@Rtrue
153       \global\byppstart@Rfalse
154     \else
155       \def\@tempb{pstart}%
156       \ifx\@tempa\@tempb
157         \global\bypage@Rfalse
158         \global\byppstart@Rtrue
159       \else
160         \def\@tempb{section}
161         \ifx\@tempa\@tempb
162           \global\bypage@Rfalse
163           \global\byppstart@Rfalse
164         \else
165           \led@warn@BadLineation
166         \fi
167       \fi
168     \fi
169   \fi}}

```

```
170 \WithSuffix\newcommand\lineation*[1]{%
```

```

171 \lineation{#1}%
172 \lineationR{#1}%
173 }%

```

`\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.

```

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

```

By default put right text numbers at the right.

```

184 \line@marginR=\@ne
185

```

`\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.

```

186 \newcounter{firstlinenumR}
187 \setcounter{firstlinenumR}{5}
188 \newcounter{linenumincrementR}
189 \setcounter{linenumincrementR}{5}

```

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

```

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

```

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

`\linenumincrement`

`\firstsublinenum`

`\sublinenumincrement`

`\firstlinenum*`

`\linenumincrement*`

`\firstsublinenum*`

`\sublinenumincrement*`

```

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

```

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

```

219 \newcommand*\Rlineflag{R}
220

```

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

```

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

```

`\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`.

```

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

```

```

235     \fi
236   \fi}
237

```

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.

```

238 \newcount\line@numR
239 \newcount\subline@numR
240 \newcount\absline@numR
241

```

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

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

```

242 \list@create{\line@listR}
243 \list@create{\insertlines@listR}
244 \list@create{\actionlines@listR}
245 \list@create{\actions@listR}
246

```

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

```

\linesinpar@listR
\maxlinesinpar@list
247 \list@create{\linesinpar@listL}
248 \list@create{\linesinpar@listR}
249 \list@create{\maxlinesinpar@list}
250

```

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

```

251 \newcount\page@numR
252

```

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.

```

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

```

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

```

254 \ifledRcol
255 \list@clear{\line@listR}%
256 \list@clear{\insertlines@listR}%
257 \list@clear{\actionlines@listR}%
258 \list@clear{\actions@listR}%
259 \list@clear{\linesinpar@listR}%
260 \list@clear{\linesonpage@listR}
261 \else
262 \list@clearing@reg
263 \list@clear{\linesinpar@listL}%
264 \list@clear{\linesonpage@listL}%
265 \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).

```

266 \list@clear{\maxlinesinpar@list}

```

Now get the file and interpret it.

```

267 \get@linelistfile{#1}%
268 \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.

```

269 \ifledRcol
270 \global\page@numR=\m@ne
271 \ifx\actionlines@listR\empty
272 \gdef\next@actionlineR{1000000}%
273 \else
274 \gl@p\actionlines@listR\to\next@actionlineR
275 \gl@p\actions@listR\to\next@actionR
276 \fi
277 \else
278 \global\page@num=\m@ne
279 \ifx\actionlines@list\empty
280 \gdef\next@actionline{1000000}%
281 \else
282 \gl@p\actionlines@list\to\next@actionline
283 \gl@p\actions@list\to\next@action
284 \fi
285 \fi}
286

```

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.

```

287 \newcommand{\@nl@regR}{%
288   \ifx\l@dchset@num\relax \else
289     \advance\absline@numR \@ne
290     \set@line@action
291     \let\l@dchset@num\relax
292     \advance\absline@numR \m@ne
293     \advance\line@numR \m@ne%    % do we need this?
294   \fi
295   \advance\absline@numR \@ne
296   \ifx\next@page@numR\relax \else
297     \page@action
298     \let\next@page@numR\relax
299   \fi
300   \ifx\sub@change\relax \else
301     \ifnum\sub@change>\z@
302       \sublines@true
303     \else
304       \sublines@false
305     \fi
306     \sub@action
307     \let\sub@change\relax
308   \fi
309   \ifcase\@lockR
310   \or
311     \@lockR \tw@
312   \or\or
313     \@lockR \z@
314   \fi
315   \ifcase\sub@lockR
316   \or
317     \sub@lockR \tw@
318   \or\or
319     \sub@lockR \z@
320   \fi
321   \ifsublines@
322     \ifnum\sub@lockR<\tw@
323       \advance\subline@numR \@ne
324     \fi
325   \else
326     \ifnum\@lockR<\tw@

```

```

327     \advance\line@numR \@ne \subline@numR \z@
328     \fi
329 \fi}
330
331 \renewcommand*{\@nl}[2]{%
332   \fix@page{#1}%
333   \ifledRcol
334     \@nl@regR
335   \else
336     \@nl@reg
337   \fi}
338

```

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

```

\fix@page
339 \newcount\last@page@numR
340 \last@page@numR=-10000
341 \renewcommand*{\fix@page}[1]{%
342   \ifledRcol
343     \ifnum #1=\last@page@numR
344     \else
345       \ifbypage@R
346         \line@numR \z@ \subline@numR \z@
347       \fi
348       \page@numR=#1\relax
349       \last@page@numR=#1\relax
350       \def\next@page@numR{#1}%
351     \fi
352   \else
353     \ifnum #1=\last@page@num
354     \else
355       \ifbypage@
356         \line@num \z@ \subline@num \z@
357       \fi
358       \page@num=#1\relax
359       \last@page@num=#1\relax
360       \def\next@page@num{#1}%
361       \listcsxadd{normal@page@break}{\the\absline@num}
362     \fi
363   \fi}
364

```

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

```

365 \renewcommand*{\@adv}[1]{%
366   \ifsublines@
367     \ifledRcol
368       \advance\subline@numR by #1\relax
369       \ifnum\subline@numR<\z@
370         \led@warn@BadAdvancelineSubline

```

```

371     \subline@numR \z@
372     \fi
373   \else
374     \advance\subline@num by #1\relax
375     \ifnum\subline@num<\z@
376       \led@warn@BadAdvancelineSubline
377       \subline@num \z@
378     \fi
379   \fi
380 \else
381   \ifledRcol
382     \advance\line@numR by #1\relax
383     \ifnum\line@numR<\z@
384       \led@warn@BadAdvancelineLine
385       \line@numR \z@
386     \fi
387   \else
388     \advance\line@num by #1\relax
389     \ifnum\line@num<\z@
390       \led@warn@BadAdvancelineLine
391       \line@num \z@
392     \fi
393   \fi
394 \fi
395 \set@line@action}
396

```

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

```

397 \renewcommand*{\@set}[1]{%
398   \ifledRcol
399     \ifsublines@
400       \subline@numR=#1\relax
401     \else
402       \line@numR=#1\relax
403     \fi
404     \set@line@action
405   \else
406     \ifsublines@
407       \subline@num=#1\relax
408     \else
409       \line@num=#1\relax
410     \fi
411     \set@line@action
412   \fi}
413

```

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

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

```

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

```

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

```

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

```

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

```

434 \renewcommand*{\set@line@action}{%
435   \ifledRcol
436     \xright@appenditem{\the\absline@numR}\to\actionlines@listR
437     \ifsublines@
438       \@l@dttempcnta=-\subline@numR
439     \else
440       \@l@dttempcnta=-\line@numR
441     \fi
442     \advance\@l@dttempcnta by -5000\relax
443     \xright@appenditem{\the\@l@dttempcnta}\to\actions@listR
444   \else
445     \xright@appenditem{\the\absline@num}\to\actionlines@list
446     \ifsublines@
447       \@l@dttempcnta=-\subline@num
448     \else
449       \@l@dttempcnta=-\line@num
450     \fi
451     \advance\@l@dttempcnta by -5000\relax
452     \xright@appenditem{\the\@l@dttempcnta}\to\actions@list
453   \fi}
454

```

`\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.

```

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

`\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.

```

472 \newcount\@lockR
473 \newcount\sub@lockR
474
475 \newcommand*{\do@lockonR}{%
476   \xright@appenditem{\the\absline@numR}\to\actionlines@listR
477   \ifsublines@
478     \xright@appenditem{-1005}\to\actions@listR
479     \ifnum\sub@lockR=\z@
480       \sub@lockR \@ne
481     \else
482       \ifnum\sub@lockR=\thr@@
483         \sub@lockR \@ne
484       \fi
485     \fi
486   \else
487     \xright@appenditem{-1003}\to\actions@listR
488     \ifnum\@lockR=\z@
489       \@lockR \@ne
490     \else
491       \ifnum\@lockR=\thr@@
492         \@lockR \@ne
493       \fi
494     \fi
495   \fi}
496
497 \renewcommand*{\do@lockon}{%

```

```

498 \ifx\next\lock@off
499   \global\let\lock@off=\skip@lockoff
500 \else
501   \ifledRcol
502     \do@lockonR
503   \else
504     \do@lockonL
505   \fi
506 \fi}

```

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

```

\do@lockoff 507
\do@lockoffR 508
\skip@lockoff 509 \newcommand{\do@lockoffR}{%
510   \xright@appenditem{\the\absline@numR}\to\actionlines@listR
511   \ifsublines@
512     \xright@appenditem{-1006}\to\actions@listR
513     \ifnum\sub@lockR=\tw@
514       \sub@lockR \thr@@
515     \else
516       \sub@lockR \z@
517     \fi
518   \else
519     \xright@appenditem{-1004}\to\actions@listR
520     \ifnum\@lockR=\tw@
521       \@lockR \thr@@
522     \else
523       \@lockR \z@
524     \fi
525   \fi}
526
527 \renewcommand*{\do@lockoff}{%
528   \ifledRcol
529     \do@lockoffR
530   \else
531     \do@lockoffL
532   \fi}
533 \global\let\lock@off=\do@lockoff
534

```

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

```

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

```

543

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

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

544 `\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.

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

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

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

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

```
568 #2
569 \else
```

And when not in right text

```
570 \@ref@reg{#1}{#2}%
571 \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.

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

`\@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.

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

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`.

```
588 \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 589 \newif\iffirst@linenum@out@R
590 \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.

```

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

```

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

```

```

685      {\if@RTL\flag@start\else\flag@end\fi% be careful on the direction of writing with polyglossia
686      }
687  }

```

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

```

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

```

`\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`.

```

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

```

```

730   \fi
731   \fi}
732

```

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 733 \newenvironment{pairs}{%}
734   \l@dpairingtrue
735   \l@dpagingfalse
736   \initnumbering@sectcmd
737 }{%
738   \l@dpairingfalse
739 }

```

The `pages` environment additionally sets the ‘column’ widths to the `\textwidth` (as known at the time the package is called). In this environment, there are two text in parallel on 2 pages. To prevent chapters starting on a lefthand page, the `\chapter` command is redefined to not clear pages.

```

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

```

```

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

```

```

758 \newif\ifinstanzaL
759 \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.

```

760 \newenvironment{Leftside}{%
761   \ledRcolfalse
762   \setcounter{pstartL}{1}
763   \let\pstart\pstartL
764   \let\thepstart\thepstartL
765   \let\pend\pendL
766   \let\memorydump\memorydumpL
767   \Leftsidehook
768   \let\old@startstanza\@startstanza
769   \def\@startstanza[##1]{\global\instanzaLtrue\old@startstanza[##1]}
770 }{
771   \Leftsidehookend}

```

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

```

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

```

Rightside The `Rightside` environment is only slightly more complicated than the `Leftside`. Apart from indicating that right text is being provided it ensures that the right right text code will be used.

```

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

```

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.

```
798 \newcount\num@linesR
```

```
799 \newbox\one@lineR
```

```
800 \newcount\par@lineR
```

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

`\pstartR`

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

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

```
801
```

```
802 \newcounter{pstartL}
```

```
803 \newcounter{pstartLold}
```

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

```
805 \newcounter{pstartR}
```

```
806 \newcounter{pstartRold}
```

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

```
808
```

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

```
  \if@nobreak%
```

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

```
  \else%
```

```

813   \let\@oldnobreak\@nobreakfalse%
814   \fi%
815   \@nobreaktrue%
816   \ifnumbering \else%
817     \led@err@PstartNotNumbered%
818     \beginnumbering%
819   \fi%
820   \ifnumberedpar@%
821     \led@err@PstartInPstart%
822   \pend%
823 \fi%

```

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

```

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

```

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

```

Dump the optional arguments

```

849 \ifstrempy{#1}%
850   {}%
851   {\csgdef{before@pstartL@the\l@dnumpstartsL}{\noindent#1}}%
852 }

853 \newcommandx*{\pstartR}[1][1]{%
854   \if@nobreak%

```

```

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

```

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

```

896 \newcommandx*{\pendL}[1][1]{%
897 \ifnumbering \else%
898 \led@err@PendNotNumbered%
899 \fi%
900 \ifnumberedpar@ \else%
901 \led@err@PendNoPstart%

```

902 \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.

```
903 \l@dzero penalties%
904 \endgraf\global\num@lines=\prevgraf\egroup%
905 \global\par@line=0%
```

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

```
906 \endgroup%
907 \ignorespaces%
908 \@oldnbreak%
909 \ifnumberpstart%
910   \addtocounter{pstartL}{1}%
911 \fi
912 \parledgroup@beforenotes@save{L}%
```

Dump content of the optional argument.

```
913 \ifstrempy{#1}%
914   {}%
915   {\csgdef{after@pendL@the\l@dnumpstartsL}{\noindent#1}}%
916 }
```

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

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

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.
939 \newbox\l@dleftbox
940 \newbox\l@drightbox
941

\countLline  We need to know the number of lines processed.
\countRline  942 \newcount\countLline
943   \countLline \z@
944 \newcount\countRline
945   \countRline \z@
946

\@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 947 \newcount\@donereallinesL
948 \newcount\@donetotallinesL
949 \newcount\@donereallinesR
950 \newcount\@donetotallinesR
951

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

952 \newcommand*\do@lineL{%
953   \advance\countLline \@ne
954   \ifvbox\namebox{\l@dLcolrawbox\the\l@dpscL}%
955   {\vbadness=10000
956    \splittopskip=\z@
957    \do@lineLhook
958    \l@demtyd@ta
959    \global\setbox\one@line=\vsplit\namebox{\l@dLcolrawbox\the\l@dpscL}
960                                     to\baselineskip}%
961   \IfStrEq{\splitfirstmarks\parledgroup@}{\begin}{\parledgroup@notes@startL}{\}
962   \unvbox\one@line \global\setbox\one@line=\lastbox
963   \getline@numL
964   \ifnum\@lock>\@ne%
965     \inserthangingsymboltrue%
966   \else%
967     \inserthangingsymbolfalse%
968   \fi
969   \setbox\l@dleftbox
970   \hb@xt@ \Lcolwidth{%

```

```

971 \affixline@num
972 \xifinlist{\the\l@dpscl}{\eled@sections@@}%
973 {}%
974 {\print@lineL}}%
975 \add@penaltiesL
976 \global\advance\@donereallinesL\@ne
977 \global\advance\@donetotallinesL\@ne
978 \else
979 \setbox\l@dleftbox \hb@xt@ \Lcolwidth{\hspace*{\Lcolwidth}}%
980 \global\advance\@donetotallinesL\@ne
981 \fi}
982
983

```

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

```

984 \def\print@lineL{%
985 \affixpstart@numL%
986 \l@dld@ta %space kept for backward compatibility
987 \add@inserts\affixside@note%
988 \l@dlsn@te %space kept for backward compatibility
989 {\ledllfill\hb@xt@ \wd\one@line{\do@insidelineLhook\inserthangingsymbolL\new@lineL\l@dunhbox@line
990 \l@drsn@te}}

```

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

```

991 \def\print@eledsectionL{%
992 \add@inserts\affixside@note%
993 \addtocounter{pstartL}{-1}%
994 \ifdefstring{\@eledsectnotoc}{L}{\ledsectnotoc}{%
995 \ifdefstring{\@eledsectmark}{L}{\ledsectnomark}%
996 \numdef{\temp@}{\l@dpscl-1}%
997 \xifinlist{\temp@}{\eled@sections@@}{\@nbreaktrue}{\@nbreakfalse}%
998 \@eled@sectioningtrue%
999 \csuse{eled@sectioning@\the\l@dpscl}%
1000 \@eled@sectioningfalse%
1001 \global\csundef{eled@sectioning@\the\l@dpscl}%
1002 \if@RTL%
1003 \hspace{-3\paperwidth}%
1004 {\hbox{\l@dunhbox@line{\one@line}} \new@line}%
1005 \else%
1006 \hspace{3\paperwidth}%
1007 {\new@line \hbox{\l@dunhbox@line{\one@line}}}%
1008 \fi%
1009 \vskip\eledsection@correcting@skip%
1010 }

```

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

```

\do@lineRhook 1011 \newcommand*{\do@lineLhook}{}
\do@insidelineLhook 1012 \newcommand*{\do@lineRhook}{}
\do@insidelineRhook 1013 \newcommand*{\do@insidelineLhook}{}

```

```
1014 \newcommand*{\do@insidelineRhook}{%
1015
```

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

```
1016 \newcommand*{\do@lineR}{%
1017   \ledRcol@true%
1018   \advance\countRline \@ne
1019   \ifvbox\namebox{\l@dRcolrawbox\the\l@dpscR}%
1020   {\vbadness=10000
1021     \splittopskip=\z@
1022     \do@lineRhook
1023     \l@demtyd@ta
1024     \global\setbox\one@lineR=\vsplit\namebox{\l@dRcolrawbox\the\l@dpscR}
1025       to\baselineskip}%
1026   \IfStrEq{\splitfirstmarks\parledgroup@}{begin}{\parledgroup@notes@startR}{%
1027     \unvbox\one@lineR \global\setbox\one@lineR=\lastbox
1028     \getline@numR
1029     \ifnum\@lockR>\@ne%
1030       \inserthangingsymbolRtrue
1031     \else%
1032       \inserthangingsymbolRfalse%
1033     \fi%
1034     \setbox\l@dtrightbox
1035     \hb@xt@ \Rcolwidth{%
1036       \affixline@numR%
1037       \xifinlist{\the\l@dpscR}{\eled@sectionsR@@}%
1038       }%
1039     {\print@lineR}%
1040   }%
1041   \add@penaltiesR
1042   \global\advance\@donereallinesR\@ne
1043   \global\advance\@donetotallinesR\@ne
1044 \else
1045   \setbox\l@dtrightbox \hb@xt@ \Rcolwidth{\hspace*{\Rcolwidth}}
1046   \global\advance\@donetotallinesR\@ne
1047 \fi
1048 \ledRcol@false%
1049 }
1050
1051
```

```
\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.

```
1052 \newcommand*{\getline@numR}{%
```

```

1053 \global\advance\absline@numR \@ne
1054 \do@actionsR
1055 \do@ballastR
1056 \ifledgroupnotesR@ \else \ifnumberline
1057 \ifsublines@
1058   \ifnum\sub@lockR<\tw@
1059     \global\advance\subline@numR \@ne
1060   \fi
1061 \else
1062   \ifnum\@lockR<\tw@
1063     \global\advance\line@numR \@ne
1064     \global\subline@numR \z@
1065   \fi
1066 \fi
1067 \fi
1068 \fi
1069 }
1070 \newcommand*{\getline@numL}{%
1071   \global\advance\absline@num \@ne
1072   \do@actions
1073   \do@ballast
1074   \ifledgroupnotesL@ \else \ifnumberline
1075   \ifsublines@
1076     \ifnum\sub@lock<\tw@
1077       \global\advance\subline@num \@ne
1078     \fi
1079   \else
1080     \ifnum\@lock<\tw@
1081       \global\advance\line@num \@ne
1082       \global\subline@num \z@
1083     \fi
1084   \fi
1085 \fi
1086 \fi
1087 }
1088
1089

```

`\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.

```

1090 \newcommand*{\do@ballastR}{\global\ballast@count=\z@
1091   \begingroup
1092     \advance\absline@numR \@ne
1093     \ifnum\next@actionlineR=\absline@numR
1094       \ifnum\next@actionR>-1001
1095         \global\advance\ballast@count by -\c@ballast
1096       \fi
1097     \fi
1098   \endgroup}

```

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

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

```

1099 \newcommand*{\do@actions@fixedcodeR}{%
1100   \ifcase\@l@tempcnta%
1101   \or%                               % 1001
1102     \global\sublines@true
1103   \or%                               % 1002
1104     \global\sublines@false
1105   \or%                               % 1003
1106     \global\@lockR=\@ne
1107   \or%                               % 1004
1108     \ifnum\@lockR=\tw@
1109       \global\@lockR=\thr@@
1110     \else
1111       \global\@lockR=\z@
1112     \fi
1113   \or%                               % 1005
1114     \global\sub@lockR=\@ne
1115   \or%                               % 1006
1116     \ifnum\sub@lockR=\tw@
1117       \global\sub@lockR=\thr@@
1118     \else
1119       \global\sub@lockR=\z@
1120     \fi
1121   \or%                               % 1007
1122     \l@dskipnumbertrue
1123   \else
1124     \led@warn@BadAction
1125   \fi}
1126
1127
1128 \newcommand*{\do@actionsR}{%
1129   \global\let\do@actions@nextR=\relax
1130   \@l@tempcntb=\absline@numR
1131   \ifnum\@l@tempcntb<\next@actionlineR\else
1132     \ifnum\next@actionR>-1001\relax
1133       \global\page@numR=\next@actionR
1134       \ifbypage@R
1135         \global\line@numR \z@ \global\subline@numR \z@
1136       \fi
1137     \else
1138       \ifnum\next@actionR<-4999\relax    % 9/05 added relax here
1139         \@l@tempcnta=-\next@actionR
1140         \advance\@l@tempcnta by -5001\relax
1141         \ifsublines@
1142           \global\subline@numR=\@l@tempcnta

```

```

1143     \else
1144         \global\line@numR=\@l@tempcnta
1145     \fi
1146     \else
1147         \@l@tempcnta=-\next@actionR
1148         \advance\@l@tempcnta by -1000\relax
1149         \do@actions@fixedcodeR
1150     \fi
1151 \fi
1152 \ifx\actionlines@listR\empty
1153     \gdef\next@actionlineR{1000000}%
1154 \else
1155     \gl@p\actionlines@listR\to\next@actionlineR
1156     \gl@p\actions@listR\to\next@actionR
1157     \global\let\do@actions@nextR=\do@actionsR
1158 \fi
1159 \fi
1160 \do@actions@nextR}
1161

```

17.4 Line number printing

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

```

\ch@cksub@l@ckR 1162
\ch@ck@l@ckR 1163 \providecommand*\l@dcalcnm}[3]{%
\fx@l@cksR 1164 \ifnum #1 > #2\relax
\affixline@numR 1165 \l@tempcnta = #1\relax
1166 \advance\l@tempcnta by -#2\relax
1167 \divide\l@tempcnta by #3\relax
1168 \multiply\l@tempcnta by #3\relax
1169 \advance\l@tempcnta by #2\relax
1170 \else
1171 \l@tempcnta=#2\relax
1172 \fi}
1173
1174 \newcommand*\ch@cksub@l@ckR}{%
1175 \ifcase\sub@lockR
1176 \or
1177 \ifnum\sublock@disp=\@ne
1178 \l@tempcntb \z@ \l@tempcnta \@ne
1179 \fi
1180 \or
1181 \ifnum\sublock@disp=\tw@
1182 \else
1183 \l@tempcntb \z@ \l@tempcnta \@ne
1184 \fi
1185 \or
1186 \ifnum\sublock@disp=\z@
1187 \l@tempcntb \z@ \l@tempcnta \@ne

```

```

1188 \fi
1189 \fi}
1190
1191 \newcommand*{\ch@ck@l@ckR}{%
1192 \ifcase\@lockR
1193 \or
1194 \ifnum\lock@disp=\@ne
1195 \@l@tempcntb \z@ \@l@tempcnta \@ne
1196 \fi
1197 \or
1198 \ifnum\lock@disp=\tw@
1199 \else
1200 \@l@tempcntb \z@ \@l@tempcnta \@ne
1201 \fi
1202 \or
1203 \ifnum\lock@disp=\z@
1204 \@l@tempcntb \z@ \@l@tempcnta \@ne
1205 \fi
1206 \fi}
1207
1208 \newcommand*{\f@x@l@cksR}{%
1209 \ifcase\@lockR
1210 \or
1211 \global\@lockR \tw@
1212 \or \or
1213 \global\@lockR \z@
1214 \fi
1215 \ifcase\sub@lockR
1216 \or
1217 \global\sub@lockR \tw@
1218 \or \or
1219 \global\sub@lockR \z@
1220 \fi}
1221
1222
1223 \newcommand*{\affixline@numR}{%
1224 \ifledgroupnotesR\else\ifnumberline
1225 \ifl@dskipnumber
1226 \global\l@dskipnumberfalse
1227 \else
1228 \ifsublines@
1229 \@l@tempcntb=\subline@numR
1230 \l@dcalcnm{\subline@numR}{\c@firstsublinenumR}{\c@sublinenumincrementR}%
1231 \ch@cksub@lockR
1232 \else
1233 \@l@tempcntb=\line@numR
1234 \ifx\linenumberlist\empty
1235 \l@dcalcnm{\line@numR}{\c@firstlinenumR}{\c@linenumincrementR}%
1236 \else
1237 \@l@tempcnta=\line@numR

```

```

1238 \edef\rem@inder{\linenumberlist,\number\line@numR,}%
1239 \edef\sc@n@list{\def\noexpand\sc@n@list
1240   ###1,\number\l@dttempcnta,###2|{\def\noexpand\rem@inder{###2}}}%
1241 \sc@n@list\expandafter\sc@n@list\rem@inder|
1242 \ifx\rem@inder\empty\advance\l@dttempcnta\@ne\fi
1243 \fi
1244 \ch@ck@l@ckR
1245 \fi
1246 \ifnum\l@dttempcnta=\l@dttempcntb
1247 \if@twocolumn
1248 \if@firstcolumn
1249 \gdef\l@dld@ta{\llap{\leftlinenumR}}}%
1250 \else
1251 \gdef\l@drd@ta{\rlap{\rightlinenumR}}}%
1252 \fi
1253 \else
1254 \l@dttempcntb=\line@marginR
1255 \ifnum\l@dttempcntb>\@ne
1256 \advance\l@dttempcntb by\page@numR
1257 \fi
1258 \ifodd\l@dttempcntb
1259 \gdef\l@drd@ta{\rlap{\rightlinenumR}}}%
1260 \else
1261 \gdef\l@dld@ta{\llap{\leftlinenumR}}}%
1262 \fi
1263 \fi
1264 \fi
1265 \f@x@l@cksR
1266 \fi
1267 \fi
1268 \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 1269
\leftpstartnumR 1270 \newcommand*{\affixpstart@numL}{%
\rightpstartnumR 1271 \ifsidepstartnum
\leftpstartnumL 1272 \if@twocolumn
\rightpstartnumL 1273 \if@firstcolumn
\ifpstartnumR 1274 \gdef\l@dld@ta{\llap{\leftpstartnumL}}}%
1275 \else

```

```

1276     \gdef\l@drd@ta{\rlap{{\rightpstartnumL}}}%
1277     \fi
1278   \else
1279     \@l@tempcntb=\line@margin
1280     \ifnum\@l@tempcntb>\@ne
1281       \advance\@l@tempcntb \page@num
1282     \fi
1283     \ifodd\@l@tempcntb
1284       \gdef\l@drd@ta{\rlap{{\rightpstartnumL}}}%
1285     \else
1286       \gdef\l@dld@ta{\llap{{\leftpstartnumL}}}%
1287     \fi
1288   \fi
1289 \fi
1290 }
1291 \newcommand*{\affixpstart@numR}{%
1292 \ifsidepstartnum
1293 \if@twocolumn
1294   \if@firstcolumn
1295     \gdef\l@dld@ta{\llap{{\leftpstartnumR}}}%
1296   \else
1297     \gdef\l@drd@ta{\rlap{{\rightpstartnumR}}}%
1298   \fi
1299 \else
1300   \@l@tempcntb=\line@marginR
1301   \ifnum\@l@tempcntb>\@ne
1302     \advance\@l@tempcntb \page@numR
1303   \fi
1304   \ifodd\@l@tempcntb
1305     \gdef\l@drd@ta{\rlap{{\rightpstartnumR}}}%
1306   \else
1307     \gdef\l@dld@ta{\llap{{\leftpstartnumR}}}%
1308   \fi
1309 \fi
1310 \fi
1311 }
1312
1313 \newcommand*{\leftpstartnumL}{%
1314 \ifpstartnum
1315 \thepstartL
1316 \kern\linenumsep\global\pstartnumfalse\fi
1317 }
1318 \newcommand*{\rightpstartnumL}{%
1319 \ifpstartnum\kern\linenumsep
1320 \thepstartL
1321 \global\pstartnumfalse\fi
1322 }
1323 \newif\ifpstartnumR
1324 \pstartnumRtrue
1325 \newcommand*{\leftpstartnumR}{%

```

```

1326 \ifpstartnumR
1327 \thepstartR
1328 \kern\linenumsep\global\pstartnumRfalse\fi
1329 }
1330 \newcommand*{\rightpstartnumR}{
1331 \ifpstartnumR\kern\linenumsep
1332 \thepstartR
1333 \global\pstartnumRfalse\fi
1334 }

```

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.

```

1335 \list@create{\inserts@listR}

```

`\add@insertsR` The right text version.

```

\add@inserts@nextR 1336 \newcommand*{\add@insertsR}{%
1337 \global\let\add@inserts@nextR=\relax
1338 \ifx\inserts@listR\empty \else
1339 \ifx\next@insertR\empty
1340 \ifx\insertlines@listR\empty
1341 \global\noteschanged@true
1342 \gdef\next@insertR{100000}%
1343 \else
1344 \glp\insertlines@listR\to\next@insertR
1345 \fi
1346 \fi
1347 \ifnum\next@insertR=\absline@numR
1348 \glp\inserts@listR\to\@insertR
1349 \@insertR
1350 \global\let\@insertR=\undefined
1351 \global\let\next@insertR=\empty
1352 \global\let\add@inserts@nextR=\add@insertsR
1353 \fi
1354 \fi
1355 \add@inserts@nextR}
1356

```

17.7 Penalties

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

In the code below, which is a virtual copy of the original `\add@penalties`, `\num@lines` is the number of lines in the whole paragraph, and `\par@line` is the

line we’re working on at the moment. The count `\@l@tempcnta` is used to calculate and accumulate the penalty; it is initially set to the value of `\ballast@count`, which has been worked out in `\do@ballast`. Finally, the penalty is checked to see that it doesn’t go below -10000 .

```
\newcommand*{\add@penaltiesR}{\@l@tempcnta=\ballast@count
\ifnum\num@linesR>\@ne
\global\advance\par@lineR \@ne
\ifnum\par@lineR=\@ne
\advance\@l@tempcnta by \clubpenalty
\fi
\@l@tempcntb=\par@lineR \advance\@l@tempcntb \@ne
\ifnum\@l@tempcntb=\num@linesR
\advance\@l@tempcnta by \widowpenalty
\fi
\ifnum\par@lineR<\num@linesR
\advance\@l@tempcnta by \interlinepenalty
\fi
\fi
\ifnum\@l@tempcnta=\z@
\relax
\else
\ifnum\@l@tempcnta>-10000
\penalty\@l@tempcnta
\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.

```
1357 \newcommand*{\add@penaltiesL}{\}
1358 \newcommand*{\add@penaltiesR}{\}
1359
```

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.

```
1360 \newcommand*{\flush@notesR}{\%
1361 \xloop
1362 \ifx\inserts@listR\empty \else
1363 \glp\inserts@listR\to\@insertR
1364 \@insertR
1365 \global\let\@insertR=\undefined
1366 \repeat}
```

1367

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:

<code>\printlinesR</code>	<code>#1</code>	<code> </code>	<code>#2</code>	<code> </code>	<code>#3</code>	<code> </code>	<code>#4</code>	<code> </code>	<code>#5</code>	<code> </code>	<code>#6</code>	<code> </code>	<code>#7</code>
<code>\printlinesR</code>	start-page		line		subline		end-page		line		subline		font

```

1368 \def\printlinesR#1|#2|#3|#4|#5|#6|#7|{\begingroup
1369   \setprintlines{#1}{#2}{#3}{#4}{#5}{#6}%
1370   \ifl@d@pnum #1\fullstop\fi
1371   \ifledplinenum \linenumr@p{#2}\Rlineflag\else \symlinenum\fi
1372   \ifl@d@ssub \fullstop \sublinenumr@p{#3}\fi
1373   \ifl@d@dash \endashchar\fi
1374   \ifl@d@pnum #4\fullstop\fi
1375   \ifl@d@elin \linenumr@p{#5}\Rlineflag\fi
1376   \ifl@d@esl \ifl@d@elin \fullstop\fi \sublinenumr@p{#6}\fi
1377 \endgroup}
1378
1379 \let\ledsavedprintlines\printlines
1380
```

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.

```

1381 \list@create{\labelref@listR}
1382
```

`\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.

```

1383 \renewcommand*{\edlabel}[1]{\@bsphack
1384   \ifledRcol
1385     \write\linenum@outR{\string\@lab}%

```

```

1386 \ifx\labelref@listR\empty
1387 \xdef\label@refs{\zz@@@}%
1388 \else
1389 \glp\labelref@listR\to\label@refs
1390 \fi
1391 \ifvmode
1392 \advancelabel@refs
1393 \fi
1394 \protected@write\@auxout{%
1395 {\string\l@dmake@labelsR\space\thepage|\label@refs|\the\c@pstartR|{#1}}%
1396 \else
1397 \write\linenum@out{\string\@lab}%
1398 \ifx\labelref@list\empty
1399 \xdef\label@refs{\zz@@@}%
1400 \else
1401 \glp\labelref@list\to\label@refs
1402 \fi
1403 \ifvmode
1404 \advancelabel@refs
1405 \fi
1406 \protected@write\@auxout{%
1407 {\string\l@dmake@labelsR\space\thepage|\label@refs|\the\c@pstartR|{#1}}%
1408 \fi
1409 \@esphack}
1410
1411

```

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

```

1412 \def\l@dmake@labelsR#1|#2|#3|#4|#5{%
1413 \expandafter\ifx\csname the@label#5\endcsname \relax\else
1414 \led@warn@DuplicateLabel{#4}%
1415 \fi
1416 \expandafter\gdef\csname the@label#5\endcsname{#1|#2\Rlineflag|#3|#4}%
1417 \ignorespaces}
1418 \AtBeginDocument{%
1419 \def\l@dmake@labelsR#1|#2|#3|#4|#5{%
1420 }
1421

```

`\@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.

```

1422 \renewcommand*{\@lab}{%
1423 \ifledRcol
1424 \xright@appenditem{\linenumr@p{\line@numR}}{%
1425 \ifsublines@ \sublinenumr@p{\subline@numR}\else 0\fi}%
1426 \to\labelref@listR
1427 \else

```

```

1428 \xright@appenditem{\linenumr@p{\line@num}}|{%
1429 \ifsublines@ \sublinenumr@p{\subline@num}\else 0\fi}%
1430 \to\labelref@list
1431 \fi}
1432

```

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* 1433 \WithSuffix\newcommand\sidenotemargin*[1]{%
1434 \l@getsidenote@margin{#1}
1435 \global\sidenote@marginR=\@l@dttempcntb
1436 \global\sidenote@margin=\@l@dttempcntb
1437 }
1438 \newcount\sidenote@marginR
1439 \global\sidenote@margin=\@ne
1440

```

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

```

1441 \newcommand*{\affixside@noteR}{%
1442 \def\sidenotecontent@{%
1443 \numgdef{\itemcount@}{0}%
1444 \def\do##1{%
1445 \ifnumequal{\itemcount@}{0}%
1446 {%
1447 \appto\sidenotecontent@{##1}}% Not print not separator before the 1st note
1448 {\appto\sidenotecontent@{\sidenotesep ##1}}%
1449 }%
1450 \numgdef{\itemcount@}{\itemcount@+1}%
1451 }%
1452 \dolistloop{\l@dcnotesnotetext}%
1453 \ifnumgreater{\itemcount@}{1}{\led@err@ManySidenotes}{}%
1454 \gdef\@templ@d{%
1455 \gdef\@templ@n{\l@dcnotesnotetext\l@dcnotesnotetext@l\l@dcnotesnotetext@r}%
1456 \ifx\@templ@d\@templ@n \else%
1457 \if@twocolumn%
1458 \if@firstcolumn%
1459 \setl@dlp@rbox{##1}{\sidenotecontent@}%
1460 \else%
1461 \setl@drp@rbox{\sidenotecontent@}%
1462 \fi%
1463 \else%
1464 \l@dttempcntb=\sidenote@marginR%
1465 \ifnum\l@dttempcntb>\@ne%
1466 \advance\l@dttempcntb by\page@numR%

```

```

1467 \fi%
1468 \ifodd\@l@dttempcntb%
1469 \setl@drp@rbox{\sidenotecontent@}%
1470 \gdef\sidenotecontent@{}%
1471 \numdef\itemcount@{0}%
1472 \dolistloop{\l@dcstotetext@l}%
1473 \ifnumgreater\itemcount@{1}{\led@err@ManyLeftnotes}{}%
1474 \setl@dlp@rbox{\sidenotecontent@}%
1475 \else%
1476 \setl@dlp@rbox{\sidenotecontent@}%
1477 \gdef\sidenotecontent@{}%
1478 \numdef\itemcount@{0}%
1479 \dolistloop{\l@dcstotetext@r}%
1480 \ifnumgreater\itemcount@{1}{\led@err@ManyRightnotes}{}%
1481 \setl@drp@rbox{\sidenotecontent@}%
1482 \fi%
1483 \fi%
1484 \fi%
1485 }
1486

```

21 Familiar footnotes

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

```

1487 \renewcommand{\l@dbfnote}[1]{%
1488 \ifnumberedpar@
1489 \gdef\@tag{#1}%
1490 \ifledRcol%
1491 \xright@appenditem{\noexpand\vl@dbfnote{\csexpandonce{\@tag}}{\@thefnmark}}%
1492 \to\inserts@listR
1493 \global\advance\insert@countR \@ne%
1494 \else%
1495 \xright@appenditem{\noexpand\vl@dbfnote{\csexpandonce{\@tag}}{\@thefnmark}}%
1496 \to\inserts@list
1497 \global\advance\insert@count \@ne%
1498 \fi
1499 \fi\ignorespaces}
1500

```

`\normalbfnoteX`

```

1501 \renewcommand{\normalbfnoteX}[2]{%
1502 \ifnumberedpar@
1503 \ifledRcol%
1504 \ifluatex
1505 \footnotelang@lua[R]%
1506 \fi
1507 \@ifundefined{xpg@main@language}{%if polyglossia

```

```

1508     {}%
1509     {\footnotelang@poly[R]}%
1510     \protected@csxdef{thisfootnote}{\csuse{thefootnote#1}}%
1511     \xright@appenditem{\noexpand\vbfnoteX{#1}{#2}{\csexpandonce{thisfootnote}}}%
1512         \to\inserts@listR
1513     \global\advance\insert@countR \@ne%
1514 \else%
1515     \ifluatex
1516         \footnotelang@lua%
1517     \fi
1518     \@ifundefined{xpg@main@language}%if polyglossia
1519     {}%
1520     {\footnotelang@poly}%
1521     \protected@csxdef{thisfootnote}{\csuse{thefootnote#1}}%
1522     \xright@appenditem{\noexpand\vbfnoteX{#1}{#2}{\csexpandonce{thisfootnote}}}%
1523         \to\inserts@list
1524     \global\advance\insert@count \@ne%
1525 \fi
1526 \fi\ignorespaces}
1527

```

22 Verse

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

```

\inserthangingsymbolL
\inserthangingsymbolR 1528 \newif\ifinserthangingsymbolR
1529 \newcommand{\inserthangingsymbolL}{%
1530 \ifinserthangingsymbol%
1531     \ifinstanzaL%
1532         \hangingsymbol%
1533     \fi%
1534 \fi}
1535 \newcommand{\inserthangingsymbolR}{%
1536 \ifinserthangingsymbolR%
1537     \ifinstanzaR%
1538         \hangingsymbol%
1539     \fi%
1540 \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 1541 \newcommand{\correcthangingL}{%
1542 \ifl@dpaging\else%

```

```

1543 \ifinstanzaL%
1544 \ifinserthangingsymbol%
1545 \hskip \@ifundefined{sza@00}{0}{\expandafter%
1546 \noexpand\csname sza@00\endcsname}\stanzaindentbase%
1547 \fi%
1548 \fi%
1549 \fi}
1550
1551 \newcommand{\correcthangingR}{%
1552 \ifl@dpaging\else%
1553 \ifinstanzaR%
1554 \ifinserthangingsymbolR%
1555 \hskip \@ifundefined{sza@00}{0}{\expandafter%
1556 \noexpand\csname sza@00\endcsname}\stanzaindentbase%
1557 \fi%
1558 \fi%
1559 \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.

```

1560 \chardef\next=\catcode'\&
1561 \catcode'\&=\active
1562

```

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

```

1563 \newenvironment{astanza}{%
1564 \startstanzahook
1565 \catcode'\&\active
1566 \global\stanza@count\@ne\stanza@modulo\@ne
1567 \ifnum\usernamecount{sza@00}=\z@
1568 \let\stanza@hang\relax
1569 \let\endlock\relax
1570 \else
1571 %%% \interlinepenalty\@M % this screws things up, but I don't know why
1572 \rightskip\z@ plus 1fil\relax
1573 \fi
1574 \ifnum\usernamecount{szp@00}=\z@
1575 \let\sza@penalty\relax
1576 \fi
1577 \def&{%
1578 \endlock\mbox{}}%
1579 \sza@penalty
1580 \global\advance\stanza@count\@ne
1581 \@astanza@line}%
1582 \def\&{%
1583 \endlock\mbox{}}
1584 \pend

```

```

1585 \endstanzaextra}%
1586 \pstart
1587 \@astanza@line
1588 }-{}
1589

```

`\@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.

```

1590 \newcommand*{\@astanza@line}{%
1591 \ifnum\value{stanzaindentrepetition}=0
1592 \parindent=\csname sza@number\stanza@count
1593 @\endcsname\stanzaindentbase
1594 \else
1595 \parindent=\csname sza@number\stanza@modulo
1596 @\endcsname\stanzaindentbase
1597 \managestanza@modulo
1598 \fi
1599 \par
1600 \stanza@hang%\mbox{}%
1601 \ignorespaces}
1602

```

Lastly reset the modified category codes.

```

1603 \catcode'\&=\next
1604

```

23 Naming macros

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

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

```

\unhnamebox 1605 \providecommand*{\newnamebox}[1]{%
\unvnamebox 1606 \expandafter\newbox\csname #1\endcsname}
\namebox 1607 \providecommand*{\setnamebox}[1]{%
1608 \expandafter\setbox\csname #1\endcsname}
1609 \providecommand*{\unhnamebox}[1]{%
1610 \expandafter\unhbox\csname #1\endcsname}
1611 \providecommand*{\unvnamebox}[1]{%
1612 \expandafter\unvbox\csname #1\endcsname}
1613 \providecommand*{\namebox}[1]{%
1614 \csname #1\endcsname}
1615

```

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

```

\usenamecount 1616 \providecommand*{\newnamecount}[1]{%

```

```

1617 \expandafter\newcount\csname #1\endcsname}
1618 \providecommand*\{usernamecount}[1]{%
1619     \csname #1\endcsname}
1620

```

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.

```

1621 \newcount\l@dc@maxchunks
1622 \newcommand*\{maxchunks}[1]{\l@dc@maxchunks=#1}
1623 \maxchunks{5120}
1624

```

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

```

\l@dnumpstartsR 1625 \newcount\l@dnumpstartsR
1626

```

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

```

\l@pscR 1627 \newcount\l@dpscL
1628 \newcount\l@dpscR
1629

```

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

```

1630 \newcommand*\{l@dsetuprawboxes}{%
1631     \@l@tempcntb=\l@dc@maxchunks
1632     \loop\ifnum\@l@tempcntb>\z@
1633         \newnamebox{\l@dLcolrawbox\the\@l@tempcntb}
1634         \newnamebox{\l@dRcolrawbox\the\@l@tempcntb}
1635         \advance\@l@tempcntb \m@ne
1636     \repeat}
1637

```

`\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.

```

1638 \newcommand*\{l@dsetupmaxlinecounts}{%
1639     \@l@tempcntb=\l@dc@maxchunks
1640     \loop\ifnum\@l@tempcntb>\z@
1641         \newnamecount{\l@dmaxlinesinpar\the\@l@tempcntb}

```

```

1642 \advance\@l@tempcntb \m@ne
1643 \repeat}
1644 \newcommand*{\l@deromaxlinecounts}{%
1645 \begingroup
1646 \l@tempcntb=\l@dc@maxchunks
1647 \loop\ifnum\@l@tempcntb>\z@
1648 \global\usenamecount{\l@maxlinesinpar\the\@l@tempcntb}=\z@
1649 \advance\@l@tempcntb \m@ne
1650 \repeat
1651 \endgroup}
1652

```

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

```

1653 \AtBeginDocument{%
1654 \l@dsetuprawboxes
1655 \l@dsetupmaxlinecounts
1656 \l@deromaxlinecounts
1657 \l@dnumpestartsL=\z@
1658 \l@dnumpestartsR=\z@
1659 \l@dpscL=\z@
1660 \l@dpscR=\z@}
1661

```

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 1662 \newif\ifl@dusedbabel
\l@dusedbabeltrue 1663 \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 `\`.

```

1664 \newcommand*{\l@dbbl@set@language}[1]{%
1665   \edef\language{#1}%
1666   \select@language{\language}%
1667   \if@filesw
1668     \protected@write\@auxout{}\string\select@language{\language}%
1669     \addtocontents{toc}{\string\select@language{\language}%
1670     \addtocontents{lof}{\string\select@language{\language}%
1671     \addtocontents{lot}{\string\select@language{\language}%
1672   \fi}
1673
```

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 1674 \providecommand{\selectlanguage}[1]{%
1675 \newcommand*{\l@duselanguage}[1]{%
1676   \gdef\theledlanguageL{}
1677   \gdef\theledlanguageR{}
1678
```

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

```

1679 \AtBeginDocument{%
1680   \ifundefined{xpg@main@language}{%
1681     \ifundefined{bbl@main@language}{%
```

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

```

1682   \l@dusedbabelfalse
1683   \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.

```

1684   \l@dusedbabeltrue
1685   \let\l@doldselectlanguage\selectlanguage
1686   \let\l@doldbbl@set@language\bbl@set@language
1687   \let\bbl@set@language\l@dbbl@set@language
1688   \renewcommand{\selectlanguage}[1]{%
1689     \l@doldselectlanguage{#1}%
1690     \ifledRcol \gdef\theledlanguageR{#1}%
1691     \else      \gdef\theledlanguageL{#1}%
1692   \fi}

```

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

```

1693 \renewcommand*{\l@duselanguage}[1]{%
1694 \l@doldselectlanguage{#1}}

    Lastly, initialise the left and right languages to the current babel one.
1695 \gdef\theledlanguageL{\bbl@main@language}%
1696 \gdef\theledlanguageR{\bbl@main@language}%
1697 }%
1698 }

    If on Polyglossia
1699 { \let\old@otherlanguage\otherlanguage%
1700 \renewcommand{\otherlanguage}[2][]{%
1701 \selectlanguage[#1]{#2}%
1702 \ifledRcol \gdef\theledlanguageR{#2}%
1703 \else \gdef\theledlanguageL{#2}%
1704 \fi}%
1705 \let\l@duselanguage\xpg@set@language%
1706 \gdef\theledlanguageL{\xpg@main@language}%
1707 \gdef\theledlanguageR{\xpg@main@language}%
1708 % \end{macrocode}
1709 % That's it.
1710 % \begin{macrocode}
1711 }}

```

26 Parallel columns

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

```

\@eledsectionR 1712 \newsavebox{\@eledsectionL}%
1713 \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.

```

1714 \newcommand*{\Columns}{%
1715 \eledsection@correcting@skip=-\baselineskip% Correction for sections' titles
1716 \setcounter{pstartL}{\value{pstartLold}}
1717 \setcounter{pstartR}{\value{pstartRold}}
1718 \ifnum\l@dnumpstartsL=\l@dnumpstartsR\else
1719 \led@err@BadLeftRightPstarts{\the\l@dnumpstartsL}{\the\l@dnumpstartsR}%
1720 \fi

```

Start a group and zero counters, etc.

```

1721 \begingroup
1722 \l@dzeropenalties
1723 \endgraf\global\num@lines=\prevgraf
1724 \global\num@linesR=\prevgraf
1725 \global\par@line=\z@
1726 \global\par@lineR=\z@
1727 \global\l@dpscL=\z@
1728 \global\l@dpscR=\z@

```

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

```
1729 \check@pstarts
1730 \loop\if@pstarts
1731 \global\pstartnumtrue
1732 \global\pstartnumRtrue
```

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

```
1733 \global\advance\l@dpscL \@ne
1734 \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

```
1735 \checkraw@text
1736 { \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.

```
1737 \l@duselanguage{\theledlanguageL}%
1738 \do@lineL
1739 \xifinlist{\the\l@dpscL}{\eled@sections@@}
1740 {%
1741 \ifdefstring{\@eledsectmark}{L}%
1742 {\csuse{eled@sectmark@the\l@dpscL}%
1743 }{}}%
1744 \global\csundef{eled@sectmark@the\l@dpscL}%
1745 \savebox{\@eledsectionL}{\parbox[t][t]{\Lcolwidth}{\vbox{}\print@eledse
1746 }%
1747 }%
1748 \l@duselanguage{\theledlanguageR}%
1749 \do@lineR
1750 \xifinlist{\the\l@dpscR}{\eled@sectionsR@@}
1751 {%
1752 \ifdefstring{\@eledsectmark}{R}%
1753 {\csuse{eled@sectmark@the\l@dpscR}%
1754 }{}}%
1755 \global\csundef{eled@sectmark@the\l@dpscR}%
1756 \savebox{\@eledsectionR}{\parbox[t][t]{\Rcolwidth}{\vbox{}\print@eledse
1757 }%
1758 }%
1759 \hb@xt@ \hsize{%
1760 \ifdefstring{\columns@position}{L}{\hfill }%
1761 \unhbox\l@dleftbox%
1762 \ifhbox\@eledsectionL%
1763 \usebox{\@eledsectionL}%
1764 \fi%
1765 \print@columnseparator%
1766 \unhbox\l@drightbox%
1767 \ifhbox\@eledsectionR%
1768 \usebox{\@eledsectionR}%
1769 \fi%
1770 }
```

```

1768         \fi%
1769         \ifdefstring{\columns@position}{R}{\hfill}%
1770     }%
1771     \checkraw@text
1772     \checkverseL
1773     \checkverseR
1774     \checkpb@columns
1775     \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.

```

1776     \@writelinesinparL
1777     \@writelinesinparR
1778     \check@pstarts
1779     \ifbypstart@
1780         \write\linenum@out{\string\@set[1]}
1781         \resetprevline@
1782     \fi
1783     \ifbypstart@R
1784         \write\linenum@outR{\string\@set[1]}
1785         \resetprevline@
1786     \fi
1787     \addtocounter{pstartL}{1}
1788     \addtocounter{pstartR}{1}
1789     \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.

```

1790     \flush@notes
1791     \flush@notesR
1792     \endgroup
1793     \global\l@dpscL=\z@
1794     \global\l@dpscR=\z@
1795     \global\l@dnumpstartsL=\z@
1796     \global\l@dnumpstartsR=\z@
1797     \ignorespaces
1798     \global\instanzaLfalse
1799     \global\instanzaRfalse}
1800

```

`\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).

```

1801 \def\print@columnseparator{%
1802     \ifdim\beforecolumnseparator<0pt%
1803         \hfill%
1804     \else%
1805         \hspace{\beforecolumnseparator}%
1806     \fi%

```

```

1807 \columnseparator%
1808 \ifdim\aftercolumnseparator<0pt%
1809   \hfill%
1810 \else%
1811   \hspace{\beforecolumnseparator}%
1812 \fi%
1813 }%
1814 %\end{macrocode}
1815 % \end{macro}
1816 % \begin{macro}{\checkpb@columns}
1817 % \cs{checkpb@columns} prevent or make pagebreaking in columns, depending of the use of \
1818 %   \begin{macrocode}
1819
1820 \newcommand{\checkpb@columns}{%
1821   \newif\if@pb
1822   \newif\if@nopb
1823   \IfStrEq{\led@pb@setting}{before}{%
1824     \numdef{\next@absline}{\the\absline@num+1}%
1825     \numdef{\next@abslineR}{\the\absline@numR+1}%
1826     \xifinlistcs{\next@absline}{l@prev@pb}{\@pbtrue}{}%
1827     \xifinlistcs{\next@abslineR}{l@prev@pbR}{\@pbtrue}{%
1828     \xifinlistcs{\next@absline}{l@prev@nopb}{\@nopbtrue}{}%
1829     \xifinlistcs{\next@abslineR}{l@prev@nopbR}{\@nopbtrue}{%
1830   }{}
1831   \IfStrEq{\led@pb@setting}{after}{%
1832     \xifinlistcs{\the\absline@num}{l@prev@pb}{\@pbtrue}{}%
1833     \xifinlistcs{\the\absline@numR}{l@prev@pbR}{\@pbtrue}{%
1834     \xifinlistcs{\the\absline@num}{l@prev@nopb}{\@nopbtrue}{}%
1835     \xifinlistcs{\the\absline@numR}{l@prev@nopbR}{\@nopbtrue}{%
1836   }{}
1837 \if@nopb\nopagebreak[4]\enlargethispage{\baselineskip}\fi
1838 \if@pb\pagebreak[4]\fi
1839 }

```

`\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).

```

1840 \newcommand*{\columnseparator}{%
1841   \smash{\rule[-0.2\baselineskip]{\columnrulewidth}{1.05\baselineskip}}}
1842 \newdimen\columnrulewidth
1843 \columnrulewidth=\z@
1844

```

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

```

1845 \newcommand*{\columnspostion}[1]{%
1846   \xdef\columns@position{#1}%
1847 }%

```

```
1848 \xdef\columns@position{R}%
```

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

```
1849 \newlength{\beforecolumnseparator}%
1850 \setlength{\beforecolumnseparator}{-2pt}%
1851
1852 \newlength{\aftercolumnseparator}%
1853 \setlength{\aftercolumnseparator}{-2pt}%
```

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

```
\@pstartstrue 1854 \newif\if@pstarts
\@pstartsfalse 1855 \newcommand*\check@pstarts}{%
\check@pstarts 1856 \@pstartsfalse
1857 \ifnum\l@dnumpstartsL>\l@dpscL
1858 \@pstartstrue
1859 \else
1860 \ifnum\l@dnumpstartsR>\l@dpscR
1861 \@pstartstrue
1862 \fi
1863 \fi
1864 }
1865
```

`\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 1866 \newif\ifaraw@text
1867 \araw@textfalse
1868 \newcommand*\checkraw@text}{%
1869 \araw@textfalse
1870 \ifvbox\namebox{\l@dLcolrawbox\the\l@dpscL}
1871 \araw@texttrue
1872 \else
1873 \ifvbox\namebox{\l@dRcolrawbox\the\l@dpscR}
1874 \araw@texttrue
1875 \fi
1876 \fi
1877 }
1878
```

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

```
1879 \newcommand*\@writelinesinparL}{%
1880 \edef\next{%
1881 \write\linenum@out{\string\@pend[\the\@donereallinesL]}}%
1882 \next
1883 \global\@donereallinesL \z@}
```

```

1884 \newcommand*{\@writelinesinparR}{%
1885   \edef\next{%
1886     \write\linenum@outR{\string\@pendR[\the\@donereallinesR]}}%
1887   \next
1888   \global\@donereallinesR \z@}
1889

```

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` 1890 `\newcount\numpagelinesL`
 1891 `\newcount\numpagelinesR`
 1892 `\newcount\l@dminpagelines`
 1893

`\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.

```

1894 \newcommand*{\Pages}{%
1895   \eledsection@correcting@skip=-2\baselineskip% line correcting for section titles.
1896   \setcounter{pstartL}{\value{pstartLold}}
1897   \setcounter{pstartR}{\value{pstartRold}}
1898   \parledgroup@notespacing@set@correction
1899   \typeout{}
1900   \typeout{***** PAGES *****}
1901   \ifnum\l@dnumpstartsL=\l@dnumpstartsR\else
1902     \led@err@BadLeftRightPstarts{\the\l@dnumpstartsL}{\the\l@dnumpstartsR}%
1903   \fi

```

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

```

1904 \cleartol@devenpage
1905 \begingroup
1906   \l@dzeropenalties
1907   \endgraf\global\num@lines=\prevgraf
1908   \global\num@linesR=\prevgraf
1909   \global\par@line=\z@
1910   \global\par@lineR=\z@
1911   \global\l@dpsscL=\z@
1912   \global\l@dpsscR=\z@
1913   \writtenlinesLfalse
1914   \writtenlinesRfalse

```

Check if there are chunks to be processed.

```

1915   \check@pstarts
1916   \loop\if@pstarts

```

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

```
1917      \global\advance\l@dpscL \@ne
1918      \global\advance\l@dpscR \@ne
```

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

```
1919      \getlinesfromparlistL
1920      \getlinesfromparlistR
1921      \l@dcalc@maxoftwo{\@cs@linesinparL}{\@cs@linesinparR}%
1922      {\usernamecount{l@dmaxlinesinpar\the\l@dpscL}}%
1923      \check@pstarts
1924      \repeat
```

Zero the counts again, ready for the next bit.

```
1925      \global\l@dpscL=\z@
1926      \global\l@dpscR=\z@
```

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

```
1927      \getlinesfrompagelistL
1928      \getlinesfrompagelistR
1929      \l@dcalc@minoftwo{\@cs@linesonpageL}{\@cs@linesonpageR}%
1930      {\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.

```
1931      \check@pstarts
1932      \if@pstarts
```

Increment the chunk counts to get the first pair.

```
1933      \global\advance\l@dpscL \@ne
1934      \global\advance\l@dpscR \@ne
```

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

```
1935      \global\@donereallinesL=\z@
1936      \global\@donetotallinesL=\z@
1937      \global\@donereallinesR=\z@
1938      \global\@donetotallinesR=\z@
```

Start a loop over the boxes (chunks).

```
1939      \checkraw@text
1940 %      \begingroup
1941 {      \loop\ifaraw@text
```

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

```
1942      \checkpageL
1943      \l@duselanguage{\theledlanguageL}%
1944 %%%      \begingroup
1945 {      \loop\ifl@dsamepage%
```

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

```

1946         \ifdefstring{\@eledsectnotoc}{L}{\ledsectnotoc}{%
1947         \csuse{before@pstartL@the\l@dpscL}
1948         \global\csundef{before@pstartL@the\l@dpscL}
1949         \do@lineL
1950         \xifinlist{the\l@dpscL}{\eled@sections@}
1951         {\print@eledsectionL}%
1952         {}%
1953         \advance\numpagelinesL \@ne
1954         \ifshiftedpstarts
1955             \ifdim\ht\l@dleftbox>Opt\hb@xt@ \hsize{\ledstrutL\unhbox\l@dleftbox}
1956         \else%
1957             \parledgroup@correction@notespacing{L}
1958             \hb@xt@ \hsize{\ledstrutL\unhbox\l@dleftbox}%
1959         \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.

```

1960         \checkpageL%
1961         \get@nextboxL%
1962         \checkverseL
1963         \checkpbl
1964         \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.

```

1965         \ifl@pagefull
1966         \@writelinesonpageL{the\numpagelinesL}%
1967         \else
1968         \@writelinesonpageL{1000}%
1969         \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.

```

1970         \numpagelinesL \z@
1971         \parledgroup@correction@notespacing@init
1972         \clearl@dleftpage }%

```

Now do the same for the right text.

```

1973         \checkpageR
1974         \l@duselanguage{theledlanguageR}%
1975 {
1976         \loop\ifl@dsamepage%
1977         \initnumbering@sectcountR
1978         \ifdefstring{\@eledsectnotoc}{R}{\ledsectnotoc}{%
1979         \csuse{before@pstartR@the\l@dpscR}

```

```

1980      \do@lineR
1981      \xifinlist{\the\l@dpscR}{\eled@sectionsR@@}
1982          {\print@eledsectionR}%
1983          {}%
1984      \advance\numpagelinesR \@ne
1985      \ifshiftedpstarts
1986          \ifdim\ht\l@drightbox>0pt\hb@xt@ \hsize{\ledstrutR\unhbox\l@drightbox}\fi%
1987      \else%
1988          \parledgroup@correction@notespacing{R}
1989          \hb@xt@ \hsize{\ledstrutR\unhbox\l@drightbox}%
1990      \fi
1991      \checkpageR%
1992      \get@nextboxR%
1993      \checkverseR
1994      \checkpbr
1995      \repeat
1996      \ifl@dpagfull
1997          \@writelinesonpageR{\the\numpagelinesR}%
1998      \else
1999          \@writelinesonpageR{1000}%
2000      \fi
2001      \numpagelinesR=\z@
2002      \parledgroup@correction@notespacing@init

```

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

```

2003      \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.

```

2004      \checkraw@text
2005      \ifaraw@text
2006          \getlinesfrompagelistL
2007          \getlinesfrompagelistR
2008          \l@dcalc@minoftwo{\@cs@linesonpageL}{\@cs@linesonpageR}%
2009                          {\l@dminpagelines}%
2010      \fi
2011      \repeat}

```

We have now output the text from all the chunks.

```

2012      \fi

```

Make sure that there are no inserts hanging around.

```

2013      \flush@notes
2014      \flush@notesR
2015      \endgroup

```

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

```

2016      \global\l@dpscL=\z@
2017      \global\l@dpscR=\z@

```

```

2018 \global\l@dnumpstartsL=\z@
2019 \global\l@dnumpstartsR=\z@
2020 \global\instanzaLfalse
2021 \global\instanzaRfalse
2022 \ignorespaces}
2023

```

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

```

\ledstrutR 2024 \newcommand*{\ledstrutL}{\strut}
2025 \newcommand*{\ledstrutR}{\strut}
2026

```

`\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@drighpage` get us onto an odd and even page, respectively, checking that we end up on the immediately next page.

```

2027 \providecommand{\cleartoevenpage}[1][\@empty]{%
2028   \clearpage
2029   \ifodd\c@page\hbox{#1}\clearpage\fi}
2030 \newcommand*{\cleartol@devenpage}{%
2031   \ifdim\pagetotal<\topskip% on an empty page
2032   \else
2033     \clearpage
2034   \fi
2035   \ifodd\c@page\hbox{}\clearpage\fi}
2036 \newcommand*{\clearl@dleftpage}{%
2037   \clearpage
2038   \ifodd\c@page\else
2039     \led@err@LeftOnRightPage
2040     \hbox{}%
2041   \cleardoublepage
2042 \fi}
2043 \newcommand*{\clearl@drighpage}{%
2044   \clearpage
2045   \ifodd\c@page
2046     \led@err@RightOnLeftPage
2047     \hbox{}%
2048   \cleartoevenpage
2049 \fi}
2050

```

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

```

\cs@linesinparR 2051 \newcommand*{\getlinesfromparlistL}{%
2052   \ifx\linesinpar@listL\empty
2053     \gdef\cs@linesinparL{0}%
2054   \else

```

```

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

```

`\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 2064 \newcommand*\getlinesfrompagelistL{%
2065 \ifx\linesonpage@listL\empty
2066 \gdef\cs@linesonpageL{1000}%
2067 \else
2068 \gl@p\linesonpage@listL\to\cs@linesonpageL
2069 \fi}
2070 \newcommand*\getlinesfrompagelistR{%
2071 \ifx\linesonpage@listR\empty
2072 \gdef\cs@linesonpageR{1000}%
2073 \else
2074 \gl@p\linesonpage@listR\to\cs@linesonpageR
2075 \fi}
2076

```

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

```

2077 \newcommand*\@writelinesonpageL}[1]{%
2078 \edef\next{\write\linenum@out{\string\@lopL{#1}}}%
2079 \next}
2080 \newcommand*\@writelinesonpageR}[1]{%
2081 \edef\next{\write\linenum@outR{\string\@lopR{#1}}}%
2082 \next}
2083

```

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

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

```

2084 \newcommand*\l@dcalc@maxoftwo}[3]{%
2085 \ifnum #2>#1\relax
2086 #3=#2\relax
2087 \else
2088 #3=#1\relax
2089 \fi}
2090 \newcommand*\l@dcalc@minoftwo}[3]{%
2091 \ifnum #2<#1\relax

```

```

2092     #3=#2\relax
2093   \else
2094     #3=#1\relax
2095   \fi}
2096

```

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

```

\checkpageL 2097 \newif\ifl@dsamepage
\checkpageR 2098   \l@dsamepagetrue
2099 \newif\ifl@dpagfull
2100
2101 \newcommand*{\checkpageL}{%
2102   \l@dpagfulltrue
2103   \l@dsamepagetrue
2104   \check@goal
2105   \ifdim\pagetotal<\ledthegoal
2106     \ifnum\numpagelinesL<\l@dminpagelines
2107       \else
2108         \l@dsamepagefalse
2109         \l@dpagfullfalse
2110       \fi
2111     \else
2112       \l@dsamepagefalse
2113       \l@dpagfulltrue
2114     \fi}
2115 \newcommand*{\checkpageR}{%
2116   \l@dpagfulltrue
2117   \l@dsamepagetrue
2118   \check@goal
2119   \ifdim\pagetotal<\ledthegoal
2120     \ifnum\numpagelinesR<\l@dminpagelines
2121       \else
2122         \l@dsamepagefalse
2123         \l@dpagfullfalse
2124       \fi
2125     \else
2126       \l@dsamepagefalse
2127       \l@dpagfulltrue
2128     \fi}
2129

```

`\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`.

```

2130 \newcommand{\checkpbL}{
2131   \IfStrEq{\led@pb@setting}{after}{
2132     \xifinlistcs{\the\absline@num}{l@prev@pb}{\l@dpagetrue\l@dsamepagefalse}{}
2133     \xifinlistcs{\the\absline@num}{l@prev@nopb}{\l@dpagetruefalse\l@dsamepagetrue}{}
2134   }{}
2135   \IfStrEq{\led@pb@setting}{before}{
2136     \numdef{\next@absline}{\the\absline@num+1}
2137     \xifinlistcs{\next@absline}{l@prev@pb}{\l@dpagetrue\l@dsamepagefalse}{}
2138     \xifinlistcs{\next@absline}{l@prev@nopb}{\l@dpagetruefalse\l@dsamepagetrue}{}
2139   }{}
2140 }
2141
2142 \newcommand{\checkpbR}{
2143   \IfStrEq{\led@pb@setting}{after}{
2144     \xifinlistcs{\the\absline@numR}{l@prev@pbR}{\l@dpagetrue\l@dsamepagefalse}{}
2145     \xifinlistcs{\the\absline@numR}{l@prev@nopbR}{\l@dpagetruefalse\l@dsamepagetrue}{}
2146   }{}
2147   \IfStrEq{\led@pb@setting}{before}{
2148     \numdef{\next@abslineR}{\the\absline@numR+1}
2149     \xifinlistcs{\next@abslineR}{l@prev@pbR}{\l@dpagetrue\l@dsamepagefalse}{}
2150     \xifinlistcs{\next@abslineR}{l@prev@nopbR}{\l@dpagetruefalse\l@dsamepagetrue}{}
2151   }{}
2152 }

```

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

```

2153 \newcommand{\checkverseL}{
2154   \ifinstanzaL
2155   \iflednopbinverse
2156     \ifinserthangingsymbol
2157       \numgdef{\prev@abslineverse}{\the\absline@num-1}
2158       \IfStrEq{\led@pb@setting}{after}{\lednopbnum{\prev@abslineverse}}{}
2159       \IfStrEq{\led@pb@setting}{before}{\ifnum\numpagelinesL<3\lednopbnum{\prev@abslineverse}\fi}{}
2160     \fi
2161   \fi
2162 \fi
2163 }
2164 \newcommand{\checkverseR}{
2165   \ifinstanzaR
2166     \iflednopbinverse
2167       \ifinserthangingsymbolR
2168         \numgdef{\prev@abslineverse}{\the\absline@numR-1}
2169         \IfStrEq{\led@pb@setting}{after}{\lednopbnumR{\prev@abslineverse}}{}
2170         \IfStrEq{\led@pb@setting}{before}{\ifnum\numpagelinesR<3\lednopbnumR{\prev@abslineverse}\fi}{}
2171       \fi
2172     \fi
2173 \fi
2174 }

```

`\ledthegoal` `\ledthegoal` is the amount of space allowed to be taken by text and footnotes on
`\goalfraction`
`\check@goal`

a page before a forced pagebreak. This can be controlled via `\goalfraction`.
`\ledthegoal` is calculated via `\check@goal`.

```

2175 \newdimen\ledthegoal
2176 \ifshiftedpstarts
2177     \newcommand*{\goalfraction}{0.95}
2178 \else
2179     \newcommand*{\goalfraction}{0.9}
2180 \fi
2181
2182 \newcommand*{\check@goal}{%
2183     \ledthegoal=\goalfraction\pagegoal}
2184

```

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

```

\ifwrittenlinesL 2185 \newif\ifwrittenlinesL
2186 \newif\ifwrittenlinesR
2187

```

`\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.

```

2188 \newcommand*{\get@nextboxL}{%
2189     \ifvbox\namebox{1@dLcolrawbox\the\1@dpscL}% box is not empty

    The current box is not empty; do nothing.

2190 \else%                                box is empty

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

2191     \ifnum\usenamecount{1@dmaxlinesinpar\the\1@dpscL}>\@donetotallinesL
2192         \parledgroup@notes@endL
2193     \else

    Sufficient lines have been output.

2194         \ifnum\usenamecount{1@dmaxlinesinpar\the\1@dpscL}=\@donetotallinesL
2195             \parledgroup@notes@endL
2196         \fi
2197     \ifwrittenlinesL\else

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

2198         \@writelinesinparL
2199         \writtenlinesLtrue
2200     \fi
2201     \ifnum\1@dnumstartsL>\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.

2202         \writtenlinesLfalse
2203         \ifbypstart@
2204             \ifnum\value{pstartL}<\value{pstartLold}
2205                 \else

```

```

2206         \global\line@num=0
2207         \resetprevline@
2208     \fi
2209 \fi
2210 % Add the content of the optional argument of the previous \cs{pend}.
2211 % \begin{macrocode}
2212     \csuse{after@pendL@the\l@dpscL}%
2213     \global\csundef{after@pendL@the\l@dpscL}%
2214 % \end{macrocode}
2215 % \begin{macrocode}
2216     \addtocounter{pstartL}{1}
2217     \global\pstartnumtrue
2218     \l@dcalc@maxoftwo{\the\usernamecount{1@dmaxlinesinpar\the\l@dpscL}}%
2219         {\the\@donetotallinesL}%
2220         {\usernamecount{1@dmaxlinesinpar\the\l@dpscL}}%
2221     \global\@donetotallinesL \z@
2222     \global\advance\l@dpscL \@ne

    Add notes of parallel ledgroup.
2223     \parledgroup@notes@endL
2224     \parledgroup@correction@notes@spacing@final{L}
2225 \else
2226 % Add the content of the optional argument of the last \cs{pend}.
2227 % \begin{macrocode}
2228     \l@dpagefulltrue
2229     \l@dsamepagefalse%
2230     \csuse{after@pendL@the\l@dpscL}%
2231     \global\csundef{after@pendL@the\l@dpscL}%
2232 % \end{macrocode}
2233 % \begin{macrocode}
2234     \fi
2235 \fi
2236 \fi}

2237 \newcommand*{\get@nextboxR}{%
2238     \ifvbox\namebox{1@dRcolrawbox\the\l@dpscR}% box is not empty
2239 \else% box is empty
2240     \ifnum\usernamecount{1@dmaxlinesinpar\the\l@dpscR}>\@donetotallinesR
2241         \parledgroup@notes@endR
2242     \else
2243         \ifnum\usernamecount{1@dmaxlinesinpar\the\l@dpscR}=\@donetotallinesR
2244             \parledgroup@notes@endR
2245         \fi
2246         \ifwrittenlinesR\else
2247             \@writelinesinparR
2248             \writtenlinesRtrue
2249         \fi
2250         \ifnum\l@dnumpstartsR>\l@dpscR
2251             \writtenlinesRfalse
2252             \ifbypstartR
2253                 \ifnum\value{pstartR}<\value{pstartRold}

```

```

2254         \else
2255             \global\line@numR=0
2256             \resetprevline@
2257         \fi
2258     \fi
2259     \csuse{after@pendR@the\l@dpscR}%
2260     \global\csundef{after@pendR@the\l@dpscR}%
2261     \addtocounter{pstartR}{1}
2262     \global\pstartnumRtrue
2263     \l@dcalc@maxoftwo{\the\usernamecount{l@dmxlinesinpar\the\l@dpscR}}%
2264             {\the\@donetotallinesR}%
2265             {\usernamecount{l@dmxlinesinpar\the\l@dpscR}}%
2266     \global\@donetotallinesR \z@
2267     \global\advance\l@dpscR \@ne
2268     \parledgroup@notes@endR
2269     \parledgroup@correction@notes@spacing@final{R}
2270 \else
2271     \l@dpagefulltrue%
2272     \l@dsamepagefalse%
2273     \csuse{after@pendR@the\l@dpscR}%
2274     \global\csundef{after@pendR@the\l@dpscR}%
2275 \fi
2276 \fi
2277 \fi}
2278

```

28 Sections' titles' commands

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

```

2279 \newcommand{\eledsectnotoc}[1]{\xdef\@eledsectnotoc{#1}}
2280 \eledsectnotoc{R}

```

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

```

2281 \newcommand{\eledsectmark}[1]{\xdef\@eledsectmark{#1}}
2282 \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`.

```

2283 \newskip\eledsection@correcting@skip

```

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

```

2284 \newwrite\eled@sectioningR@out

```

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

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

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

```
2285 \def\l@prev@pbR{}
2286 \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.

```
2287 \newcommand{\ledpbR}{\write\linenum@outR{\string\led@pbR}}
2288 \newcommand{\ledpbnumR}[1]{\write\linenum@outR{\string\led@pbnumR{#1}}}
2289 \newcommand{\lednopbR}{\write\linenum@outR{\string\led@nopbR}}
2290 \newcommand{\lednopbnumR}[1]{\write\linenum@outR{\string\led@nopbnumR{#1}}}
```

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

```
2291 \newcommand{\led@pbR}{\listcsxadd{\prev@pbR}{\the\absline@numR}}
2292 \newcommand{\led@pbnumR}[1]{\listcsxadd{\prev@pbR}{#1}}
2293 \newcommand{\led@nopbR}{\listcsxadd{\prev@nopbR}{\the\absline@numR}}
2294 \newcommand{\led@nopbnumR}[1]{\listcsxadd{\prev@nopbR}{#1}}
```

30 Parallel ledgroup

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

```
2295 \newmarks\parledgroup@
2296 \newmarks\parledgroup@series
2297 \newmarks\parledgroup@type
```

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

```
2298 \newcommand{\parledgroup@notes@startL}{%
2299   \ifnum\usenamecount{\l@maxlinesinpar\the\l@dpscl}>0%
2300   \IfStrEq{\splitfirstmarks\parledgroup@type}{footnoteX}{\csuse{bhooknoteX@\splitfirstmarks\parledgroup@type}}{\csuse{hooknoteX@\splitfirstmarks\parledgroup@type}}
2301   \IfStrEq{\splitfirstmarks\parledgroup@type}{Xfootnote}{\csuse{bhookXnote@\splitfirstmarks\parledgroup@type}}{\csuse{hookXnote@\splitfirstmarks\parledgroup@type}}
2302   \fi%}
```

```

2303 \global\ledgroupnotesL@true%
2304 \insert@noterule@ledgroup{L}%
2305 }
2306 \newcommand{\parledgroup@notes@startR}{%
2307 \ifnum\usenamecount{1@dmaxlinesinpar\the1@dpscR}>0%
2308 \IfStrEq{\splitfirstmarks\parledgroup@type}{footnoteX}{\csuse{bhooknoteX@\splitfirstma
2309 \IfStrEq{\splitfirstmarks\parledgroup@type}{Xfootnote}{\csuse{bhookXnote@\splitfirstma
2310 \fi%
2311 \global\ledgroupnotesR@true%
2312 \insert@noterule@ledgroup{R}%
2313 }

```

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

```

2314 \newcommand{\parledgroup@notes@endL}{%
2315 \global\ledgroupnotesL@false%
2316 }
2317 \newcommand{\parledgroup@notes@endR}{%
2318 \global\ledgroupnotesR@false%
2319 }

```

`\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`

```

2320 \newcommand{\insert@noterule@ledgroup}[1]{
2321 \IfStrEq{\splitbotmarks\parledgroup@}{begin}{%
2322 \IfStrEq{\splitbotmarks\parledgroup@type}{Xfootnote}{
2323 \csuse{ifledgroupnotes#1@}
2324 \vskip\skip\csuse{mp\splitbotmarks\parledgroup@series footins}
2325 \csuse{\splitbotmarks\parledgroup@series footnoterule}
2326 \fi
2327 }
2328 {}
2329 \IfStrEq{\splitbotmarks\parledgroup@type}{footnoteX}{
2330 \csuse{ifledgroupnotes#1@}
2331 \vskip\skip\csuse{mpfootins\splitbotmarks\parledgroup@series}
2332 \csuse{footnoterule\splitbotmarks\parledgroup@series}
2333 \fi
2334 }{}
2335 }
2336 {}
2337 }

```

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

```

2338 \newcommand{\parledgroupnotespadding}{}

```

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

```

2339 \dimdef{\parledgroup@notespacing@correction}{0pt}
2340 \newcommand{\parledgroup@notespacing@set@correction}{%
2341   {\notefontsetup\parledgroupnotespacing\dimgdef{\temp@spacing}{\baselineskip}}%
2342   \dimgdef{\parledgroup@notespacing@correction}{\baselineskip-\temp@spacing}%
2343 }

```

`\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.

```

2344 \newcommand{\parledgroup@correction@notespacing@init}{
2345   \dimdef{\parledgroup@notespacing@correction@accumulated}{0pt}
2346   \dimdef{\parledgroup@notespacing@correction@modulo}{0pt}
2347 }
2348 \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.

```

2349 \newcommand{\parledgroup@correction@notespacing@final}[1]{
2350   \ifparledgroup
2351     \vspace{\parledgroup@notespacing@correction@accumulated}
2352     \parledgroup@correction@notespacing@init%
2353     \ifstrequal{#1}{L}{
2354       \numdef{\@checking}{\the\l@dpscL-1}
2355     }{
2356       \numdef{\@checking}{\the\l@dpscR-1}
2357     }
2358     \dimdef{\@beforenotes@current@diff}{\csuse{@parledgroup@beforenotes@\@checking L}-\csuse{@parledgroup@beforenotes@\@checking R}}
2359     \ifstrequal{#1}{L}{%
2360       {% Left
2361         \ifdimgreater{\@beforenotes@current@diff}{0pt}{\vspace{-\@beforenotes@current@diff}}%
2362       }%
2363       {% Right
2364         \ifdimgreater{\@beforenotes@current@diff}{0pt}{\vspace{\@beforenotes@current@diff}}%
2365       }%
2366     } \fi
2367 }

```

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

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

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

```

2368 {}
2369 \newcommand{\parledgroup@correction@notespacing}[1]{%
2370     \csuse{ifledgroupnotes#1@}%
2371     \vspace{-\parledgroup@notespacing@correction}%
2372     \dimdef{\parledgroup@notespacing@correction@accumulated}{\parledgroup@notespacing@correction}
2373     \dimdef{\parledgroup@notespacing@correction@modulo}{\parledgroup@notespacing@correction}
2374     \ifdimless{\parledgroup@notespacing@correction@modulo}{\baselineskip}{\advance\num
2375     \dimdef{\parledgroup@notespacing@correction@modulo}{\parledgroup@notespacing@correction}
2376     }% mean greater than equal
2377     \fi%
2378 }
```

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

```

2379 \dimdef\parledgroup@beforenotesL{0pt}
2380 \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.

```

2381 \newcommand{\parledgroup@beforenotes@save}[1]{
2382     \ifparledgroup
2383         \csdimgdef{@parledgroup@beforenotes@the\csuse{1@dnumstarts#1}#1}{\csuse{parledgroup@beforenotes@save#1}}
2384         \csdimgdef{parledgroup@beforenotes#1}{0pt}
2385     \fi
2386 }
```

31 The End

i/codej

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>	365, 630, 631
<code>\&</code>	1560, 1561, 1565, 1582, 1603	<code>\@afterindentfalse</code> 755
<code>\@M</code>	1571	<code>\@arabic</code> 221, 222, 804, 807

[illegible]

- \advanceline 629, 660
 - \affixline@num 971
 - \affixline@numR 1036, 1162
 - \affixpstart@numL 985, 1269
 - \affixpstart@numR 1269
 - \affixside@note 987, 992
 - \affixside@noteR 1441
 - \aftercolumnseparator . . . 4, 1808, 1849
 - \appto 1447, 1448
 - \araw@textfalse 1866
 - \araw@texttrue 1866
 - astanza (environment) 8, 1563
 - \AtBeginDocument . . . 1418, 1653, 1679
- B**
- \ballast@count 1090, 1095
 - \bbl@main@language 1695, 1696
 - \bbl@set@language 1686, 1687
 - \beforecolumnseparator
..... 4, 1802, 1805, 1811, 1849
 - \beginnumbering 7, 779, 818
 - \beginnumberingR . . . 42, 122, 779, 862
 - \bfseries 804, 807
 - \bypage@Rfalse 142, 157, 162
 - \bypage@Rtrue 142, 152
 - \bypstart@Rfalse 142, 153, 163
 - \bypstart@Rtrue 142, 158
- C**
- \c@ballast 1095
 - \c@chapter 104
 - \c@chapterR 104
 - \c@firstlinenumR 186, 1235
 - \c@firstsublinenumR 190, 1230
 - \c@linenumincrementR 186, 1235
 - \c@page 613, 615, 2029, 2035, 2038, 2045
 - \c@pstart 1407
 - \c@pstartL 804
 - \c@pstartR 807, 1395
 - \c@section 105
 - \c@sectionR 105
 - \c@sublinenumincrementR . . 190, 1230
 - \c@subsection 106
 - \c@subsectionR 106
 - \c@subsubsection 107
 - \c@subsubsectionR 107
 - \ch@ck@l@ckR 1162
 - \ch@cksub@l@ckR 1162
 - \ch@cksub@lockR 1231
 - \chapter 741, 742, 751
 - \chapterinpages 733, 742, 753
 - \chardef 1560
 - \check@goal 2104, 2118, 2175
 - \check@pstarts
1729, 1778, 1854, 1915, 1923, 1931
 - \checkpageL 1942, 1960, 2097
 - \checkpageR 1973, 1991, 2097
 - \checkpb@columns . . . 1774, 1816, 1820
 - \checkpbL 1963, 2130
 - \checkpbR 1994, 2130
 - \checkraw@text
..... 1735, 1771, 1866, 1939, 2004
 - \checkverseL 1772, 1962, 2153
 - \checkverseR 1773, 1993, 2153
 - \cleardoublepage 2041
 - \clearl@dleftpage 1972, 2027
 - \clearl@drightrightpage 2003, 2027
 - \cleartoevenpage 2027
 - \cleartol@devenpage 1904, 2027
 - \closeout 95, 594, 600, 603, 607
 - \columnrulewidth 4, 1840
 - \Columns 3, 1714
 - \columns@position . . 1759, 1769, 1845
 - \columnseparator 4, 1807, 1840
 - \columnssposition 4, 1845
 - \correcthangingL 989, 1541
 - \correcthangingR 1541
 - \countLline 942, 953
 - \countRline 942, 1018
 - \critext 665
 - \cs 1817, 2210, 2226
 - \csdimgdef 2383, 2384
 - \csexpandonce . . 1491, 1495, 1511, 1522
 - \csgdef 851, 894, 915, 936
 - \csundef 1001, 1744, 1755,
1948, 1979, 2213, 2231, 2260, 2274
 - \csuse 999,
1510, 1521, 1742, 1753, 1947,
1978, 2212, 2230, 2259, 2273,
2300, 2301, 2308, 2309, 2323–
2325, 2330–2332, 2358, 2370, 2383
- D**
- \DeclareOption 8–10
 - \def@tempb 160
 - \dimdef 2339, 2345, 2346,
2358, 2372, 2373, 2375, 2379, 2380
 - \dimen 616, 617, 621–623, 627
 - \dimgdef 2341, 2342
 - \divide 1167

- \do@actions 1072
 - \do@actions@fixedcodeR 1099
 - \do@actions@nextR 1099
 - \do@actionsR 1054, 1099
 - \do@ballast 1073
 - \do@ballastR 1055, 1090
 - \do@insidelineLhook 989, 1011
 - \do@insidelineRhook 1011
 - \do@lineL 952, 1738, 1949
 - \do@lineLhook 957, 1011
 - \do@lineR 1016, 1749, 1980
 - \do@lineRhook 1011, 1022
 - \do@lockoff 507
 - \do@lockoffL 531
 - \do@lockoffR 507
 - \do@lockon 472
 - \do@lockonL 504
 - \do@lockonR 472
 - \dolistloop 1452, 1472, 1479
 - \dummy@ref 553
- E**
- \edfont@info 716, 719, 725, 728
 - \edlabel 1383
 - \edtext 688
 - \eled@sectioningR@out .. 70, 95, 2284
 - \eled@sections@@ 972, 997, 1739, 1950
 - \eled@sectionsR@@ 67, 1037, 1750, 1981
 - \eledpar@error 21, 30, 33, 35
 - \eledsection@correcting@skip ...
..... 1009, 1715, 1895, 2283
 - \eledsectmark 13, 2281
 - \eledsectnotoc 13, 2279
 - \empty .. 80, 83, 271, 279, 678, 701,
714, 723, 827, 871, 1152, 1234,
1242, 1338–1340, 1351, 1362,
1386, 1398, 2052, 2058, 2065, 2071
 - \end@lemmas 678, 679, 701, 702
 - \endashchar 1373
 - \endgraf 904, 925, 1723, 1907
 - \endline@num 560, 566
 - \endlock 649, 1569, 1578, 1583
 - \endnumbering 7, 73, 126, 780
 - \endnumberingR 45, 73, 110, 121, 134, 780
 - \endpage@num 559, 566
 - \endstanzaextra 1585
 - \endsub 616
 - \endsubline@num 561, 567
 - \enlargethispage 1837
- environments:
- astanza 8, 1563
 - Leftside 5, 760
 - pages 4, 733
 - pairs 3, 733
 - Rightside 5, 777
- \extensionchars 62, 116, 131, 139
- F**
- \f@x@l@cksR 1162
 - \first@linenum@out@Rfalse .. 589, 595
 - \first@linenum@out@Rtrue 589
 - \firstlinenum 5, 195
 - \firstlinenum* 5, 195
 - \firstsublinenum 5, 195
 - \firstsublinenum* 5, 195
 - \fix@page 332, 339
 - \flag@end
..... 616, 674, 684, 685, 697, 707, 708
 - \flag@start
..... 616, 673, 674, 685, 696, 697, 708
 - \flush@notes 1790, 2013
 - \flush@notesR 1360, 1791, 2014
 - \footnotelang@lua 1505, 1516
 - \footnotelang@poly 1509, 1520
 - \fullstop . 234, 1370, 1372, 1374, 1376
- G**
- \get@linelistfile 267
 - \get@nextboxL 1961, 2188
 - \get@nextboxR 1992, 2188
 - \getline@numL 963, 1070
 - \getline@numR 1028, 1052
 - \getlinesfrompagelistL
..... 1927, 2006, 2064
 - \getlinesfrompagelistR
..... 1928, 2007, 2064
 - \getlinesfromparlistL ... 1919, 2051
 - \getlinesfromparlistR ... 1920, 2051
 - \gl@p 274, 275,
282, 283, 679, 702, 718, 727,
1155, 1156, 1344, 1348, 1363,
1389, 1401, 2055, 2061, 2068, 2074
 - \goalfraction 5, 2175
- H**
- \hangingsymbol 10, 1532, 1538
 - \hb@xt@ 970, 979, 989, 1035,
1045, 1758, 1955, 1958, 1986, 1989

\hsize 845,
888, 1758, 1955, 1958, 1986, 1989

I

\if@filesw 1667
\if@firstcolumn 1248, 1273, 1294, 1458
\if@ledgroup 599
\if@nobreak 810, 854
\if@noeled@sec 68, 94
\if@nopb 1822, 1837
\if@pb 1821, 1838
\if@pstarts 1730, 1854, 1916, 1932
\if@RTL 674, 685, 697, 708, 1002
\ifaraw@text ... 1736, 1866, 1941, 2005
\ifautopar 837, 881
\ifbypage@ 355
\ifbypage@R 142, 345, 1134
\ifbypstart@ 574, 1779, 2203
\ifbypstart@R ... 142, 578, 1783, 2252
\ifdefstring 994, 995,
1741, 1752, 1759, 1769, 1946, 1977
\ifdim ... 617, 621, 623, 627, 1802,
1808, 1955, 1986, 2031, 2105, 2119
\ifdimgreater 2361, 2364
\ifdimless 2374
\iffirst@linenum@out@R 589, 593
\ifhbox 1761, 1766
\ifinserthangingsymbol
..... 1530, 1544, 2156
\ifinserthangingsymbolR
..... 1528, 1536, 1554, 2167
\ifinstanzaL 758, 758, 1531, 1543, 2154
\ifinstanzaR 758, 759, 1537, 1553, 2165
\ifl@d@dash 1373
\ifl@d@elin 1375, 1376
\ifl@d@esl 1376
\ifl@d@pnum 1370, 1374
\ifl@d@ssub 1372
\ifl@d@pagefull 1965, 1996, 2097
\ifl@d@paging 12, 1542, 1552
\ifl@d@pairing 12, 77
\ifl@d@samelang 1664
\ifl@d@samepage 1945, 1975, 2097
\ifl@d@skipnumber 1225
\ifl@d@usedbabel 1662
\iflabelpstart 847, 890
\ifledgroupnotesL@ 1074
\ifledgroupnotesR@ 1056, 1224
\iflednopbinverse 2155, 2166
\ifledplinenum 1371

\ifledRcol 12, 178,
200, 204, 208, 212, 254, 269,
333, 342, 367, 381, 398, 415,
427, 435, 456, 501, 528, 537,
546, 618, 624, 630, 637, 645,
650, 654, 659, 669, 692, 713,
1384, 1423, 1490, 1503, 1690, 1702
\ifluatex 1504, 1515
\ifnoteschanged@ 87
\ifnumberedpar@
... 820, 864, 900, 921, 1488, 1502
\ifnumbering 147, 816, 897
\ifnumberingR ... 43, 74, 112, 860, 918
\ifnumberline 1056, 1074, 1224
\ifnumberpstart ... 838, 882, 909, 930
\ifnumequal 1445
\ifnumgreater 1453, 1473, 1480
\ifodd 1258, 1283,
1304, 1468, 2029, 2035, 2038, 2045
\ifparledgroup 2350, 2382
\ifpst@rtedL 38, 824
\ifpst@rtedR 38, 868
\ifpstartnum 1314, 1319
\ifpstartnumR 1269
\ifshiftedpstarts 5, 1954, 1985, 2176
\ifsidepstartnum 839, 883, 1271, 1292
\ifstreempty 849, 892, 913, 934
\IfStrEq 961, 1026, 1823,
1831, 2131, 2135, 2143, 2147,
2158, 2159, 2169, 2170, 2300,
2301, 2308, 2309, 2321, 2322, 2329
\ifstrequal 2353, 2359
\ifsublines@ 232,
321, 366, 399, 406, 437, 446,
458, 465, 477, 511, 565, 567,
1057, 1075, 1141, 1228, 1425, 1429
\ifvbox 954, 1019, 1870, 1873, 2189, 2238
\ifvmode 1391, 1403
\ifwrittenlinesL 2185, 2197
\ifwrittenlinesR 2186, 2246
\initnumbering@sectcmd 118, 736, 745
\initnumbering@sectcountR 66, 99, 1976
\InputIfFileExists 69
\insert@count 543, 670, 693, 1497, 1524
\insert@countR 544, 669, 692, 1493, 1513
\insert@noterule@ledgroup
..... 2304, 2312, 2320
\inserthangingsymbolfalse 967
\inserthangingsymbolL ... 989, 1528
\inserthangingsymbolR 1528

- \inserthangingsymbolRfalse 1032
 - \inserthangingsymbolRtrue 1030
 - \inserthangingsymboltrue 965
 - \insertlines@listR
 - 80, 242, 256, 549, 1340, 1344
 - \inserts@list 826, 1496, 1523
 - \inserts@listR 870, 1335,
 - 1338, 1348, 1362, 1363, 1492, 1512
 - \istanzaLfalse 1798, 2020
 - \istanzaLtrue 769
 - \istanzaRfalse 1799, 2021
 - \istanzaRtrue 792
 - \interlinepenalty 1571
 - \itemcount@ 1443, 1445,
 - 1450, 1453, 1471, 1473, 1478, 1480
- L**
- \l@d@nums 716, 719, 725, 728
 - \l@d@set 414, 645, 646
 - \l@dbbl@set@language 1664, 1687
 - \l@dbfnote 1487
 - \l@dc@maxchunks 832, 834,
 - 876, 878, 1621, 1631, 1639, 1646
 - \l@dcalc@maxoftwo
 - 1921, 2084, 2218, 2263
 - \l@dcalc@minoftwo .. 1929, 2008, 2084
 - \l@dcalcnun 1162
 - \l@dchecklang 1664
 - \l@dchset@num 288, 291, 414
 - \l@dcsnotetext 1452, 1455
 - \l@dcsnotetext@l 1455, 1472
 - \l@dcsnotetext@r 1455, 1479
 - \l@emptyd@ta 958, 1023
 - \l@dend@stuff 63, 117, 132, 140
 - \l@dgetline@margin 176
 - \l@dgetsidenote@margin 1434
 - \l@dld@ta 986,
 - 1249, 1261, 1274, 1286, 1295, 1307
 - \l@dleftbox
 - .. 939, 969, 979, 1760, 1955, 1958
 - \l@dlinenumR 224
 - \l@dlsn@te 988
 - \l@dmake@labels 1407
 - \l@dmake@labelsR 1395, 1412
 - \l@dminpagelines
 - 1890, 1930, 2009, 2106, 2120
 - \l@dnumpstartsL
 - 831, 832, 834, 836, 851,
 - 915, 1625, 1657, 1718, 1719,
 - 1795, 1857, 1901, 1902, 2018, 2201
 - \l@dnumpstartsR
 - .. 47, 875, 876, 878, 880, 894,
 - 936, 1625, 1658, 1718, 1719,
 - 1796, 1860, 1901, 1902, 2019, 2250
 - \l@doldbbl@set@language 1686
 - \l@doldselectlanguage 1685, 1689, 1694
 - \l@dpagfullfalse
 - 2097, 2133, 2138, 2145, 2150
 - \l@dpagfulltrue 2097,
 - 2132, 2137, 2144, 2149, 2228, 2271
 - \l@dpagingfalse 14, 735, 750
 - \l@dpagingtrue 744
 - \l@dpairingfalse 12, 738, 749
 - \l@dpairingtrue 734, 743
 - \l@dpscL 954,
 - 959, 972, 996, 999, 1001, 1627,
 - 1659, 1727, 1733, 1739, 1742,
 - 1744, 1793, 1857, 1870, 1911,
 - 1917, 1922, 1925, 1933, 1947,
 - 1948, 1950, 2016, 2189, 2191,
 - 2194, 2201, 2212, 2213, 2218,
 - 2220, 2222, 2230, 2231, 2299, 2354
 - \l@dpscR 1019, 1024, 1037,
 - 1628, 1660, 1728, 1734, 1750,
 - 1753, 1755, 1794, 1860, 1873,
 - 1912, 1918, 1926, 1934, 1978,
 - 1979, 1981, 2017, 2238, 2240,
 - 2243, 2250, 2259, 2260, 2263,
 - 2265, 2267, 2273, 2274, 2307, 2356
 - \l@drd@ta 989,
 - 1251, 1259, 1276, 1284, 1297, 1305
 - \l@drightbox
 - 939, 1034, 1045, 1765, 1986, 1989
 - \l@drsn@te 990
 - \l@dsamepagefalse 2097,
 - 2132, 2137, 2144, 2149, 2229, 2272
 - \l@dsamepagetrue
 - 2097, 2133, 2138, 2145, 2150
 - \l@dsetupmaxlinecounts .. 1638, 1655
 - \l@dsetuprawboxes 1630, 1654
 - \l@dskipnumberfalse 1226
 - \l@dskipnumbertrue 1122
 - \l@dunhbox@line 989, 1004, 1007
 - \l@dusedbabelfalse 1662, 1682
 - \l@dusedbabeltrue 1662, 1684
 - \l@duselanguage
 - 1674, 1737, 1748, 1943, 1974
 - \l@dzeromaxlinecounts ... 1638, 1656
 - \l@dzeropenalties 903, 924, 1722, 1906
 - \l@prev@nopbR 54, 2286

- \l@prev@pbR 53, 2285
- \l@pscL 1627
- \l@pscR 1627
- \label@refs
 - 1387, 1389, 1395, 1399, 1401, 1407
- \labelref@list 1398, 1401, 1430
- \labelref@listR 1381, 1386, 1389, 1426
- \language name .. 1665, 1666, 1668–1671
- \last@page@num 353, 359
- \last@page@numR 339
- \lastbox 962, 1027
- \lastskip 616, 622
- \lcolwidth
 - . 4, 5, 16, 746, 845, 970, 979, 1745
- \led@err@BadLeftRightPstarts ...
 - 29, 1719, 1902
- \led@err@LeftOnRightPage ... 32, 2039
- \led@err@LineationInNumbered ... 148
- \led@err@ManyLeftnotes 1473
- \led@err@ManyRightnotes 1480
- \led@err@ManySidenotes 1453
- \led@err@NumberingNotStarted ... 91
- \led@err@numberingShouldHaveStarted
 - 120
- \led@err@NumberingStarted 44
- \led@err@PendNoPstart 901, 922
- \led@err@PendNotNumbered ... 898, 919
- \led@err@PstartInPstart ... 821, 865
- \led@err@PstartNotNumbered . 817, 861
- \led@err@RightOnLeftPage ... 32, 2046
- \led@err@TooManyPstarts
 - 24, 26, 833, 877
- \led@mess@NotesChanged 88
- \led@mess@SectionContinued
 - 115, 130, 138
- \led@nopbnumR 2290, 2291
- \led@nopbR 2289, 2291
- \led@pb@setting
 - 1823, 1831, 2131, 2135,
 - 2143, 2147, 2158, 2159, 2169, 2170
- \led@pbnumR 2288, 2291
- \led@pbR 2287, 2291
- \led@warn@BadAction 1124
- \led@warn@BadAdvancelineLine 384, 390
- \led@warn@BadAdvancelineSubline .
 - 370, 376
- \led@warn@BadLineation 165
- \led@warn@BadSetline 635
- \led@warn@BadSetlinenum 643
- \led@warn@DuplicateLabel 1414
- \ledgroupnotesL@false 2315
- \ledgroupnotesL@true 2303
- \ledgroupnotesR@false 2318
- \ledgroupnotesR@true 2311
- \ledllfill 989
- \lednopb 788
- \lednopbnum 2158, 2287
- \lednopbnumR 2169, 2287
- \lednopbR 788, 2289
- \ledpb 787
- \ledpbnum 2159
- \ledpbnumR 2170, 2287
- \ledpbR 787, 2287
- \ledRcol@false 1048
- \ledRcol@true 1017
- \ledRcolfalse 15, 761, 794
- \ledRcoltrue 778
- \ledrlfill 989
- \ledsavedprintlines 7, 1368
- \ledsectnomark 995
- \ledsectnotoc 994, 1946, 1977
- \ledstrutL 1955, 1958, 2024
- \ledstrutR 1986, 1989, 2024
- \ledthegoal 2105, 2119, 2175
- \leftlinenumR 224, 1249, 1261
- \leftpstartnumL 1269
- \leftpstartnumR 1269
- Leftside (environment) 5, 760
- \Leftsidehook 767, 772
- \Leftsidehookend 771, 772
- \line@list 723, 727
- \line@list@stuff 131
- \line@list@stuffR .. 62, 116, 139, 591
- \line@listR . 83, 242, 255, 567, 714, 718
- \line@margin 181, 1279
- \line@marginR 174, 1254, 1300
- \line@num . 356, 388, 389, 391, 409,
 - 420, 421, 449, 574, 1081, 1428, 2206
- \line@numR 55,
 - 231, 238, 293, 327, 346, 382,
 - 383, 385, 402, 416, 417, 440,
 - 560, 564, 578, 1063, 1135, 1144,
 - 1233, 1235, 1237, 1238, 1424, 2255
- \lineation 170, 171, 789
- \lineation* 6, 170
- \lineationR 6, 146, 172, 789
- \linenum@out
 - . 613, 619, 625, 631, 638, 646,
 - 651, 655, 1397, 1780, 1881, 2078

- \linenum@outR
 - . 588, 594, 596, 600, 601, 603,
 - 604, 607, 608, 615, 618, 624,
 - 630, 637, 645, 650, 654, 659,
 - 1385, 1784, 1886, 2081, 2287–2290
 - \linenumberlist 1234, 1238
 - \linenumincrement 5, 195
 - \linenumincrement* 5, 195
 - \linenummargin 174
 - \linenumr@p 1371, 1375, 1424, 1428
 - \linenumrepR 221, 231
 - \linenumsep
 - . 226, 228, 1316, 1319, 1328, 1331
 - \linesinpar@listL
 - 247, 263, 575, 2052, 2055
 - \linesinpar@listR
 - 247, 259, 579, 2058, 2061
 - \linesonpage@listL 264, 583, 2065, 2068
 - \linesonpage@listR 260, 586, 2071, 2074
 - \list@clear
 - . 255–260, 263, 264, 266, 826, 870
 - \list@clearing@reg 262
 - \list@create
 - ... 242–245, 247–249, 1335, 1381
 - \listcsxadd 361, 2291–2294
 - \lock@disp 1194, 1198, 1203
 - \lock@off 498, 499, 507, 654, 655
 - \lock@on 650, 651
- M**
- \managestanza@modulo 1597
 - \maxchunks 3, 1621
 - \maxlinesinpar@list 247, 266
 - \memorydump 7, 766, 783
 - \memorydumpL 125, 766
 - \memorydumpR 125, 783
 - \message 61
 - \multiply 1168
- N**
- \n@num 535, 659
 - \n@num@reg 541
 - \namebox 954, 959, 1019,
 - 1024, 1605, 1870, 1873, 2189, 2238
 - \NeedsTeXFormat 2
 - \new@line 1004, 1007
 - \new@lineL 612, 989
 - \new@lineR 614
 - \newbox 799, 939, 940, 1606
 - \newcommandx 809, 853, 896, 917
 - \newcounter 99–102, 186,
 - 188, 190, 192, 802, 803, 805, 806
 - \newif 5,
 - 13, 39, 142, 143, 589, 758, 759,
 - 1323, 1528, 1662, 1821, 1822,
 - 1854, 1866, 2097, 2099, 2185, 2186
 - \newlength 1849, 1852
 - \newmarks 2295–2297
 - \newnamebox 1605, 1633, 1634
 - \newnamecount 1616, 1641
 - \newsavebox 1712, 1713
 - \newwrite 588, 2284
 - \next@absline
 - 1824, 1826, 1828, 2136–2138
 - \next@abslineR
 - 1825, 1827, 1829, 2148–2150
 - \next@action 283
 - \next@actionline 280, 282
 - \next@actionlineR
 - . 272, 274, 1093, 1131, 1153, 1155
 - \next@actionR 275, 1094,
 - 1132, 1133, 1138, 1139, 1147, 1156
 - \next@insert 827
 - \next@insertR
 - 871, 1339, 1342, 1344, 1347, 1351
 - \next@page@num 360, 432
 - \next@page@numR 59, 296, 298, 350, 429
 - \no@expands 667, 690
 - \noindent 851, 894, 915, 936
 - \normal@page@breakR 52
 - \normal@pars 76, 830, 874
 - \normalbfnoteX 1501
 - \notefontsetup 2341
 - \noteschanged@true
 - 81, 84, 715, 724, 1341
 - \num@lines 904, 1723, 1907
 - \num@linesR 798, 925, 1724, 1908
 - \numberedpar@true 846, 889
 - \numberingRfalse 75
 - \numberingRtrue 49, 110, 135
 - \numberingtrue 127
 - \numberpstartfalse 8
 - \numberpstarttrue 8
 - \numdef 996, 1471, 1478,
 - 1824, 1825, 2136, 2148, 2354, 2356
 - \numgdef 1443, 1450, 2157, 2168
 - \numlabfont 231
 - \numpagelinesL 1890,
 - 1953, 1966, 1970, 2106, 2159, 2374

- `\numpagelinesR`
 1890, 1984, 1997, 2001, 2120, 2170
- O**
- `\old@otherlanguage` 1699
`\old@startstanza` .. 768, 769, 791, 792
`\oldchapter` 741, 751
`\one@line` ... 959, 962, 989, 1004, 1007
`\one@lineR` 798, 1024, 1027
`\openout` 70, 596, 601, 604, 608
`\otherlanguage` 1699, 1700
- P**
- `\p@pstartL` 848
`\p@pstartR` 891
`\PackageError` 21
`\page@action` 297, 426, 554
`\page@num` 278, 358, 1281
`\page@numR` 251, 270, 348,
 559, 564, 1133, 1256, 1302, 1466
`\pagebreak` 1838
`\pagegoal` 2183
`\Pages` 4, 1894
`pages` (environment) 4, 733
`\pagetotal` 2031, 2105, 2119
`pairs` (environment) 3, 733
`\paperwidth` 1003, 1006
`\par@line` 905, 1725, 1909
`\par@lineR` 798, 926, 1726, 1910
`\parbox` 1745, 1756
`\parledgroup@` .. 961, 1026, 2295, 2321
`\parledgroup@beforenotes@save` ..
 912, 933, 2381
`\parledgroup@beforenotesL` 2379
`\parledgroup@beforenotesR` 2379
`\parledgroup@correction@notespacing`
 1957, 1988, 2368
`\parledgroup@correction@notespacing@final`
 2224, 2269, 2349
`\parledgroup@correction@notespacing@init`
 1971, 2002, 2344, 2352
`\parledgroup@notes@endL`
 2192, 2195, 2223, 2314
`\parledgroup@notes@endR`
 2241, 2244, 2268, 2317
`\parledgroup@notes@startL`
 961, 2298, 2314
`\parledgroup@notes@startR`
 1026, 2298, 2314
- `\parledgroup@notespacing@correction`
 2339, 2371–2373
`\parledgroup@notespacing@correction@accumulated`
 2345, 2351, 2372
`\parledgroup@notespacing@correction@modulo`
 2346, 2373–2375
`\parledgroup@notespacing@set@correction`
 1898, 2339
`\parledgroup@series`
 2296, 2300, 2301,
 2308, 2309, 2324, 2325, 2331, 2332
`\parledgroup@type` 2297,
 2300, 2301, 2308, 2309, 2322, 2329
`\parledgroupnotespacing` .. 2338, 2341
`\parledgroupseries@` 2295
`\parledgrouptrue` 10
`\parledgrouptype@` 2295
`\pausenumbering` 781
`\pausenumberingR` 109, 781
`\pend` 5, 765, 786, 822, 1584
`\pendL` 765, 896
`\pendR` 786, 866, 917
`\prev@abslineverse`
 2157–2159, 2168–2170
`\prev@nopbR` 2285
`\prev@pbR` 2285
`\prevgraf`
 904, 925, 1723, 1724, 1907, 1908
`\print@columnseparator` .. 1764, 1801
`\print@eledsectionL`
 984, 991, 1745, 1951
`\print@eledsectionR` .. 1052, 1756, 1982
`\print@lineL` 974, 984
`\print@lineR` 1039, 1052
`\printlines` 1379
`\printlinesR` 7, 1368
`\ProcessOptions` 11
`\protected@csxdef` 1510, 1521
`\protected@edef` 847, 890
`\protected@write` ... 1394, 1406, 1668
`\ProvidesPackage` 3
`\pst@rtedLfalse` 38
`\pst@rtedLtrue` 128, 828
`\pst@rtedRfalse` 40, 48, 78
`\pst@rtedRtrue` 113, 136, 872
`\pstart` 5, 27, 31, 763, 785, 1586
`\pstartL` 763, 801
`\pstartnumfalse` 1316, 1321
`\pstartnumRfalse` 1328, 1333
`\pstartnumRtrue` 1324, 1732, 2262

- `\pstartnumtrue` 1731, 2217
`\pstartR` 785, 801
- R**
- `\Rcolwidth` 4,
 5, 16, 747, 888, 1035, 1045, 1756
`\read@linelist` 253, 592
`\rem@inder` 1238, 1240–1242
`\resetprevline@` 1781, 1785, 2207, 2256
`\resumenumbering` 782
`\resumenumberingR` 109, 782
`\rightlinenumR` 224, 1251, 1259
`\rightpstartnumL` 1269
`\rightpstartnumR` 1269
Rightside (environment) 5, 777
`\Rightsidehook` 772, 790
`\Rightsidehookend` 772, 795
`\rlap` 1251, 1259, 1276, 1284, 1297, 1305
`\Rlineflag` 7, 219, 231, 1371, 1375, 1416
`\rule` 1841
- S**
- `\savebox` 1745, 1756
`\sc@n@list` 1239, 1241
`\secdef` 756
`\section@num` 129–131
`\section@numR` 36,
 50, 61, 62, 69, 70, 114–116, 137–139
`\select@language` ... 1666, 1668–1671
`\selectlanguage` 1674
`\set@line` 668, 691, 712
`\set@line@action`
 290, 395, 404, 411, 434, 556
`\setl@dlp@rbox` 1459, 1474, 1476
`\setl@drp@rbox` 1461, 1469, 1481
`\setline` 633
`\setlinenum` 641
`\setnamebox` 836, 880, 1605
`\setprintrlines` 1369
`\shiftedpstartsfalse` 7
`\shiftedpstartstrue` 6, 8, 9
`\shiftedversesfalse` 7
`\shiftedversestrue` 6
`\showlemma` 677, 700
`\sidenote@margin` 1436, 1439
`\sidenote@marginR` 1433, 1464
`\sidenotecontent@`
 . 1442, 1447, 1448, 1459, 1461,
 1469, 1470, 1474, 1476, 1477, 1481
`\sidenotemargin` 1433
- `\sidenotemargin*` 1433
`\sidenotesep` 1448
`\skip` 2324, 2331
`\skip@lockoff` 499, 507
`\skipnumbering` 8, 658
`\skipnumbering@reg` 662
`\smash` 1841
`\splitbotmarks` 2321,
 2322, 2324, 2325, 2329, 2331, 2332
`\splitfirstmarks`
 961, 1026, 2300, 2301, 2308, 2309
`\splittopskip` 956, 1021
`\stanza@count` 1566, 1580, 1592
`\stanza@hang` 1568, 1600
`\stanza@modulo` 1566, 1595
`\stanzaindentbase`
 1546, 1556, 1593, 1596
`\startlock` 649
`\startstanzahook` 1564
`\startsub` 616
`\sub@action` 306, 455, 555
`\sub@change` 60, 300, 301, 307
`\sub@lock` 1076
`\sub@lockR` 57, 315, 317, 319,
 322, 473, 479, 480, 482, 483,
 513, 514, 516, 1058, 1114, 1116,
 1117, 1119, 1175, 1215, 1217, 1219
`\sub@off` 624, 625
`\sub@on` 618, 619
`\subline@num` 233, 356, 374,
 375, 377, 407, 447, 1077, 1082, 1429
`\subline@numR` 234,
 238, 323, 327, 346, 368, 369,
 371, 400, 438, 561, 565, 1059,
 1064, 1135, 1142, 1229, 1230, 1425
`\sublinenumincrement` 5, 195
`\sublinenumincrement*` 5, 195
`\sublinenumrep` . 1372, 1376, 1425, 1429
`\sublinenumrepR` 221, 234
`\sublines@false` 58, 304, 1104
`\sublines@true` 302, 1102
`\sublock@disp` 1177, 1181, 1186
`\symplinenum` 1371
`\sza@penalty` 1575, 1579
- T**
- `\temp@` 996, 997
`\temp@spacing` 2341, 2342
`\textwidth` 17, 19, 746, 747
`\theledlanguageL` ... 1674, 1737, 1943

<code>\theledlanguageR</code> . . . 1674, 1748, 1974	W
<code>\thepage</code> 613, 615, 1395, 1407	<code>\W</code> 2284
<code>\thepstart</code> 764, 784	<code>\wd</code> 989
<code>\thepstartL</code>	<code>\WithSuffix</code> 170, 215–218, 1433
. 8, 764, 804, 841, 848, 1315, 1320	<code>\writtenlinesLfalse</code> 1913, 2202
<code>\thepstartR</code>	<code>\writtenlinesLtrue</code> 2199
. 8, 784, 807, 884, 891, 1327, 1332	<code>\writtenlinesRfalse</code> 1914, 2251
<code>\thr@@</code> . . . 482, 491, 514, 521, 1109, 1117	<code>\writtenlinesRtrue</code> 2248
<code>\topskip</code> 2031	
U	X
<code>\unhbox</code> 1610,	<code>\x@lemma</code> 679–681, 702–704
1760, 1765, 1955, 1958, 1986, 1989	<code>\xifinlist</code> 972,
<code>\unhnamebox</code> 1605	997, 1037, 1739, 1750, 1950, 1981
<code>\unvbox</code> 962, 1027, 1612	<code>\xifinlistcs</code> 1826–
<code>\unvnamebox</code> 1605	1829, 1832–1835, 2132, 2133,
<code>\usebox</code> 1762, 1767	2137, 2138, 2144, 2145, 2149, 2150
<code>\usenamecount</code> 1567, 1574, 1616, 1648,	<code>\xpg@main@language</code> 1706, 1707
1922, 2191, 2194, 2218, 2220,	<code>\xpg@set@language</code> 1705
2240, 2243, 2263, 2265, 2299, 2307	<code>\xright@appenditem</code>
V 428, 429, 431, 432,
<code>\value</code> 825, 869, 1591,	436, 443, 445, 452, 457, 459,
1716, 1717, 1896, 1897, 2204, 2253	461, 464, 466, 468, 476, 478,
<code>\vbadness</code> 955, 1020	487, 510, 512, 519, 538, 539,
<code>\vbfnoteX</code> 1511, 1522	549, 563, 575, 579, 583, 586,
<code>\vbox</code> 836, 880, 1745, 1756	1424, 1428, 1491, 1495, 1511, 1522
<code>\vl@dbfnote</code> 1491, 1495	Z
<code>\vsplit</code> 959, 1024	<code>\zz@@@</code> 1387, 1399

Change History

v0.1	bering by pstart (like in eledmac 0.15). 41
General: First public release 1	Lineation can be by pstart (like in eledmac 0.15). 18
v0.10	New management of hangingsymbol insertion, preventing undesirable insertions. 56
General: <code>\edlabel</code> commands on the right side are now correctly indicated. 1	Prevent shift of column separator when a verse is hanged 56
<code>\edlabel</code> commands which start a paragraph are now put in the right place. 1	
v0.11	<code>\affixline@numR:</code> Changed
General: Change <code>\do@lineL</code> and <code>\do@lineR</code> to allow line num-	<code>\affixline@numR</code> to allow to disable line numbering (like in

eledmac 0.15).	46	<code>\l@dnumpstartsR</code> :	Moved
<code>\Columns</code> : Line numbering by		<code>\l@dnumpstartsL</code> to eledmac .	59
<code>pstart</code>	64	<code>\ledsavedprintlines</code> :	Simpli-
<code>\get@nextboxR</code> : Change <code>\get@nextboxL</code>		fied <code>\printlinesR</code> by using	
and <code>\get@nextboxR</code> to allow to		<code>\setprintlines</code>	52
disable line numbering (like in		<code>\ledstrutR</code> : Added <code>\ledtrutL</code> and	
eledmac 0.15).	75	<code>\ledstrutR</code>	71
<code>Pstart</code> number can be printed in		<code>\normalbfnoteX</code> :	Removed
side	75	extraneous spaces from	
v0.12		<code>\normalbfnoteX</code>	55
General: New new management of		<code>\Pages</code> : Added <code>\ledstrutL</code> to	
<code>hangingsymbol</code> insertion, pre-		<code>\Pages</code>	69
venting undesirable insertions.	56	Added <code>\ledstrutR</code> to <code>\Pages</code> .	69
v0.2		<code>\Rightsidehookend</code> :	Added
General: Added section of babel re-		<code>\Leftsidehook</code> , <code>\Leftsidehookend</code> ,	
lated code	60	<code>\Rightsidehook</code> and <code>\Rightsidehookend</code>	
Fix babel problems	1	36
<code>\Columns</code> : Added <code>\l@dchecklang</code>		<code>\sublinenumrepR</code> :	Added
and <code>\l@duselanguage</code> to		<code>\linenumrepR</code> and <code>\sublinenumrepR</code>	
<code>\Columns</code>	63	20
<code>\Pages</code> : Added <code>\l@duselanguage</code>		v0.3.a	
to <code>\Pages</code>	68	General: Minor <code>\linenummargin</code>	
v0.3		fix	1
General: Added <code>\do@lineLhook</code>		v0.3.b	
and <code>\do@lineRhook</code>	42	General: Improved parallel page	
Reorganize for ledarab	1	balancing	1
<code>\affixline@numR</code> :	Changed	v0.3.c	
<code>\affixline@numR</code> to match new		General: Compatibilty with Poly-	
eledmac	46	glossia	1
<code>\do@actions@nextR</code> :	Used	v0.3a	
<code>\do@actions@fixedcode</code> in		<code>\line@marginR</code> : Don't just	
<code>\do@actionsR</code>	45	set <code>\line@marginR</code> in	
<code>\do@lineL</code> : Added <code>\do@lineLhook</code>		<code>\linenummargin</code>	19
to <code>\do@lineL</code>	41	v0.3b	
Simplified <code>\do@lineL</code> by using		<code>\Pages</code> : Added <code>\l@dminpagelines</code>	
macros for some common code	41	calculation for succeeding page	
<code>\do@lineR</code> : Changed <code>\do@lineR</code>		pairs	70
similarly to <code>\do@lineL</code>	43	v0.4	
Leftside : Added hooks into Left-		General: No more ledparpatch. All	
side environment	36	patches are now in the main	
<code>\flag@end</code> : Removed extraneous		file.	1
spaces from <code>\flag@end</code>	31	v0.5	
<code>\ifledRcol</code> : Moved <code>\ifl@dpairing</code>		General: Corrections about	
to eledmac	14	<code>\section</code> and other titles in	
<code>\ifpst@rtedR</code> : Moved <code>\ifpst@rtedL</code>		numbered sections	1
to eledmac	15	v0.6	
<code>\l@dlinumR</code> :	Simplified	General: Be able to us <code>\chapter</code> in	
<code>\leftlinenumR</code> and <code>\rightlinenumR</code>		parallel pages.	1
by introducing <code>\l@dlinumR</code>	20		

v0.7	General: Option ‘shiftedverses’ which make there is no blank between two parallel verses with inequal length.	1	v1.1.1	<code>\pstartR</code> : Correct <code>\pstartR</code> bug introduced by 1.1.	37
v0.8	General: Possibility to have a symbol on each hanging of verses, like in the french typography. Redefine the commande <code>\hangingsymbol</code> to define the character.	1	v1.1.2	<code>\affixside@noteR</code> : Remove spurious space between line number and line content	54
v0.9	General: Possibility to number <code>\pstart</code>	8	v1.2	General: Support for <code>\led<section></code> commands in parallel texts. . .	1
	Possibility to number the pstart with the commands <code>\numberpstarttrue</code>	1	v1.2.1	<code>\initnumbering@sectcountR</code> : For the right section, the counter is defined only once.	16
	<code>\ifledRcol</code> : Moved <code>\iflledRcol</code> and <code>\ifnumberingR</code> to <code>eledmac</code>	14	v1.3	General: Manage RTL language. .	33
v0.9.1	General: The numbering of the pstarts restarts on each <code>\beginnumbering</code>	1	v1.3.2	General: Debug with some classes. .	1
v0.9.2	General: Debug : with <code>\Columns</code> , the hanging indentation now runs on the left columns and the hanging symbol is shown only when <code>\stanza</code> is used.	1	v1.3.3	General: Debugging the left notes of the right column.	54
v0.9.3	General: <code>\thepstartL</code> and <code>\thepstartR</code> use now <code>\bfseries</code> and not <code>\bf</code> , which is deprecated and makes conflicts with memoir class.	1	v1.3.4	<code>\l@dbfnote</code> : Spurious space with footnote in right column. . . .	55
v1.0	General: Compatibility with <code>eledmac</code> . Change name to <code>eledpar</code> . .	1	v1.3.5	General: Allow use of commands in sidenotes, as introduced by <code>eledmac</code> 1.0.	54
	Debug in lineation by <code>pstart</code> . .	18	v1.4	General: Added <code>\do@insidelineLhook</code> and <code>\do@insidelineRhook</code> . . .	42
v1.0.1	General: Correction on <code>\numberonlyfirstline</code> with lineation by <code>pstart</code> or by page.	1	v1.4.1	<code>\normalbfnoteX</code> : Allows one to redefine <code>\thefootnoteX</code> with <code>alph</code> when some packages are loaded.	55
v1.1	General: Shiftedverses becomes shiftedpstarts.	1		<code>\normalbfnoteX</code> : Fix bug with normal familiar footnotes when mixing RTL and LTR text. . .	55
	<code>\pstartR</code> : Add <code>\labelpstarttrue</code> (from <code>eledmac</code>).	37		<code>astanza</code> : Enable the use of <code>stanzaindentrepetition</code> within <code>astanza</code> environment.	57
				<code>\line@list@stuffR</code> : Open / close immediatly the line-list file when in minipage, except if the minipage is a ledgroup.	31
			v1.4.3	General: Corrects a false hanging verse when a verse is exactly the	

length of a line.	1	New sectioning commands, as in	
<code>\inserthangingsymbolR</code> : Hang		eledmac.	13
verse is now not automatically		<code>\Columns</code> : Modify <code>\Columns</code> to en-	
flush right.	56	able to add section's title. . . .	62
<code>\pendL</code> : Spurious spaces in <code>\pendL</code> .	39	Suppress <code>\l@dchecklang</code> from	
<code>\pendR</code> : Spurious spaces in		<code>\Columns</code>	63
<code>\pstartR</code>	40	<code>\l@dchecklang</code> : Suppress	
<code>\pstartR</code> : Spurious spaces in		<code>\l@dchecklang</code> which didn't	
<code>\pstartL</code> and <code>\pstartR</code>	37	work and was not logical, be-	
v1.5.0		cause both columns could have	
General: Add, as in eledmac, fea-		the same language but not the	
tures to manage page breaks. . .	1	main language of the docu-	
<code>\sublinenumincrement*</code> : Add		ment.	60
starred version of <code>\firstlinenum</code> ,		<code>\Pages</code> : Modify <code>\Pages</code> to enable to	
<code>\linenumincrement</code> , <code>\firstsublinenum</code> ,		add section's title.	67
<code>\sublinenumincrement</code> to		<code>\pendL</code> : As in eledmac, <code>\pendL</code> can	
change both Left and Right-		have an optional argument. . .	39
side.	19	<code>\pendR</code> : As in eledmac, <code>\pendR</code> can	
v1.6.0		have an optional argument. . .	40
General: Add tool and documenta-		<code>\print@columnseparator</code> : Move	
tion for parallel ledgroups . . .	10	some code of <code>\Columns</code> to	
v1.7.0		<code>\print@columnseparator</code> . . .	64
General: Add, as in eledmac, fea-		<code>\pstartR</code> : As in eledmac, <code>\pendL</code>	
tures to make crossrefs with		and <code>\pendR</code> can have an op-	
<code>pstart</code> numbers.	1	tional argument.	37
v1.8.0		<code>\sidenotemargin*</code> : <code>\sidenotemargin</code>	
General: <code>\beginnumbering</code> is de-		is now directly defined in eled-	
fined only on eledmac, not on		mac to be able to manage eled-	
<code>eledpar</code>	15	par.	54
<code>\l@dlsnote</code> , <code>\l@drsnote</code> and		Add <code>\sidenotemargin*</code>	54
<code>\l@dcsnote</code> defined only one		<code>\theledlanguageR</code> : Correct	
time, in eledmac.	54	left/right language setting with	
Add <code>\beforecolumnseparator</code>		<code>polyglossia</code>	62
and <code>\aftercolumnseparator</code> . . .	4	v1.8.1	
Add <code>\columnspan</code>	4	<code>\do@lineL</code> : Fix a bug with critical	
Add, as in eledmac, new system		notes at the beginning of a page,	
of sectioning commands.	1	(maybe added by v1.8.0) (?). .	41
Add, as in eledmac, possibility to		<code>\do@lineR</code> : Fix a bug with critical	
insert something after <code>\pends</code> /		notes at the beginning of a page,	
verses.	1	added by v1.8.0 (?).	43
Add, as in eledmac, possibil-		v1.8.2	
ity to insert something between		General: Debug <code>\eledxxx</code> with	
<code>\pstarts</code> / verse.	1	some paper size	1
Change <code>\do@lineR</code> and		Debug left and side note (bugs	
<code>\do@lineR</code> to allow new sec-		added by 1.8.0)	1
tioning commands.	41	<code>\eledpar@error</code> : Errors specific to	
Compatibility with <code>musixtex</code> . . .	1	eledpar send to eledpar hand-	
Debug eledmac section-		book	14
ing command after using		<code>\flag@end</code> : <code>\flag@start</code> and	
<code>\resumenumbering</code>	1	<code>\flag@end</code> are now defined only	

one time for eledmac and eled-	eledmac	1
par		31
\lineation*: Add \lineation* .		18
v1.8.3	\Pages: Debug blank pages when	
General: Add \noeledxxx, as in	using optional argument in the	
	last \pend.	67