

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

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

Abstract

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

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

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

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

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

Contents

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

*This file (**eledpar.dtx**) has version number v1.9.1, last revised 2014/09/30.

[†]herries dot press at earthlink dot net

[‡]maieul at maieul dot net

5 Left and right texts	7
6 Numbering text lines and paragraphs	9
7 Verse	10
8 Side notes	12
9 Parallel ledgroups	12
9.1 Parallel ledgroups and <code>setspace</code> package	14
10 Sectioning commands	14
11 Implementation overview	14
12 Preliminaries	14
12.1 Messages	15
13 Sectioning commands	16
14 Line counting	19
14.1 Choosing the system of lineation	19
14.2 Line-number counters and lists	22
14.3 Reading the line-list file	23
14.4 Commands within the line-list file	24
14.5 Writing to the line-list file	32
15 Marking text for notes	34
16 Parallel environments	36
17 Paragraph decomposition and reassembly	38
17.1 Boxes, counters, <code>\pstart</code> and <code>\pend</code>	38
17.2 Processing one line	42
17.3 Line and page number computation	45
17.4 Line number printing	48
17.5 Pstart number printing in side	50
17.6 Add insertions to the vertical list	52
17.7 Penalties	52
17.8 Printing leftover notes	53
18 Footnotes	53
18.1 Normal footnote formatting	53
19 Cross referencing	54
20 Side notes	56

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

List of Figures

1 Introduction

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

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

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

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

2 The `eledpar` package

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

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

2.1 General

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

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

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

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

```
\maxchunks{5120}
```

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

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

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

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

3 Parallel columns

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

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

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

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

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

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

4 Facing pages

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

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

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

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

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

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

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

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

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

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

5 Left and right texts

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

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

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

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

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

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

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

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

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

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

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

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

6 Numbering text lines and paragraphs

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

```
\beginnumbering
<text>
```

```
\endnumbering
```

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

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

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

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

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

Its default definition is:

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

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

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

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

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

7 Verse

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

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

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

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

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

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

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

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

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

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

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

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

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

```

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

```

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

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

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

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

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

8 Side notes

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

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

9 Parallel ledgroups

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

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

See the following example:

```

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

```

```

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

```

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

```

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

```

9.1 Parallel ledgroups and `setspace` package

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

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

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

10 Sectioning commands

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

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

11 Implementation overview

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

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

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

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

12 Preliminaries

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

```

1 (*code)
2 \NeedsTeXFormat{LaTeX2e}
3 \ProvidesPackage{eledpar}[2014/09/30 v1.9.1 eledmac extension for parallel texts]%
4

```

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

`\ifshiftedpstarts`

```

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

```

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

```

11 \DeclareOption{widthliketwocolumns}{\widthliketwocolumnstrue}%

```

```

12 \ProcessOptions%

```

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

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

```

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

```

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

```

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

```

12.1 Messages

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

`\eledpar@error`

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

`\led@err@BadLeftRightPstarts`

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

`\led@err@LeftOnRightPage`

`\led@err@RightOnLeftPage`

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

13 Sectioning commands

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

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

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

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

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

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

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



```

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

```

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

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

```

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

```

```

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

```

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

```

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

```

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

```

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

```

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

```

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

```

```

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

```

14 Line counting

14.1 Choosing the system of lineation

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

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

`\bypstart@Rtrue`

`\bypstart@Rfalse`

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

`\ifbypage@R`

`\bypage@Rtrue`

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

```

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

```

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

```

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

```

```

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

```

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

```

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

```

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

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

```

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

```

By default put right text numbers at the right.

```

185 \line@marginR=\@ne
186

```

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

```

187 \newcounter{firstlinenumR}

```

```

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

```

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

```

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

```

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

```

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

```

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

```

220 \newcommand*\Rlineflag{R}
221

```

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

```

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

```

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

```

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

14.2 Line-number counters and lists

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

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

```

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

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

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

`\actionlines@listR`

`\actions@listR`

```

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

`\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`

```

248 \list@create{\linesinpar@listL}
```

```

249 \list@create{\linesinpar@listR}
250 \list@create{\maxlinesinpar@list}
251

```

\page@numR The right text page number.

```

252 \newcount\page@numR
253

```

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.

```

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

```

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

```

255 \ifledRcol
256 \list@clear{\line@listR}%
257 \list@clear{\insertlines@listR}%
258 \list@clear{\actionlines@listR}%
259 \list@clear{\actions@listR}%
260 \list@clear{\linesinpar@listR}%
261 \list@clear{\linesonpage@listR}
262 \else
263 \list@clearing@reg
264 \list@clear{\linesinpar@listL}%
265 \list@clear{\linesonpage@listL}%
266 \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).

```

267 \list@clear{\maxlinesinpar@list}

```

Now get the file and interpret it.

```

268 \get@linelistfile{#1}%
269 \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.

```

270 \ifledRcol
271 \global\page@numR=\m@ne
272 \ifx\actionlines@listR\empty
273 \gdef\next@actionlineR{1000000}%
274 \else
275 \gl@p\actionlines@listR\to\next@actionlineR
276 \gl@p\actions@listR\to\next@actionR
277 \fi
278 \else

```

```

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

```

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.

```

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

```



```

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

```

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

```

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

```

```

361     \def\next@page@num{#1}%
362     \listcsxadd{normal@page@break}{\the\absline@num}
363   \fi
364 \fi}
365

```

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

```

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

```

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

```

398 \renewcommand*{\@set}[1]{%
399   \ifledRcol
400     \ifsublines@
401       \subline@numR=#1\relax
402     \else
403       \line@numR=#1\relax

```

```

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

```

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

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

```

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

```

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

```

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

```

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

```

435 \renewcommand*{\set@line@action}{%
436 \ifledRcol
437 \xright@appenditem{\the\absline@numR}\to\actionlines@listR
438 \ifsublines@
439 \l@dtempcnta=-\subline@numR
440 \else
441 \l@dtempcnta=-\line@numR
442 \fi

```

```

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

```

`\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.

```

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

```

`\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.

```

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

```

```

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

```

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

```

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

```

```

533 \fi}
534 \global\let\lock@off=\do@lockoff
535

```

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

```

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

```

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

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

```

545 \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.

```

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

```

553 \begingroup
554 \let\@ref=\dummy@ref
555 \let\page@action=\relax
556 \let\sub@action=\relax
557 \let\set@line@action=\relax

```

```

558 \let\@lab=\relax
559 #2
560 \global\endpage@num=\page@numR
561 \global\endline@num=\line@numR
562 \global\endsubline@num=\subline@numR
563 \endgroup

```

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

```

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

```

569 #2
570 \else

```

And when not in right text

```

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

```

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

```

`\@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.

```

582 \providecommand*\@lopL}[1]{}
583 \renewcommand*\@lopL}[1]{}%
584 \xright@appenditem{#1}\to\linesonpage@listL}
585 \providecommand*\@lopR}[1]{}
586 \renewcommand*\@lopR}[1]{}%
587 \xright@appenditem{#1}\to\linesonpage@listR}
588

```

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`.

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

```
591 \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.

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

`\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.

```
613 \newcommand*{\new@lineL}{%
```

```
614   \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.

```
615 \newcommand*{\new@lineR}{%
```

```
616   \write\linenum@outR{\string\@nl[\the\c@page][\thepage]}}
```


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

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

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

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

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

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

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

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

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

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

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

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

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

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.

```
666 \long\def\critext#1#2/{\leavevmode
667   \begingroup
668     \renewcommand{\@tag}{\no@expands #1}%

```

```

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

```

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

```

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

```

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

```

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

```

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 734 \newenvironment{pairs}{%}
735   \l@dpairingtrue
736   \l@dpagingfalse
737   \initnumbering@sectcmd
738   \at@begin@pairs%
739 }{%
740   \l@dpairingfalse
741 }
742
\AtBeginPairs The \AtBeginPairs macro just define a \at@begin@pairs macro, called at the
beginning of each pairs environments.
743 \newcommand{\AtBeginPairs}[1]{\xdef\at@begin@pairs{#1}}%
744 \def\at@begin@pairs{}%
745

```

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.

```

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

```

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

```

764 \newif\ifinstanzaL
765 \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.

```

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

```

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

```

\Rightsidehook 778 \newcommand*{\Leftsidehook}{}
\Rightsidehookend 779 \newcommand*{\Leftsidehookend}{}
780 \newcommand*{\Rightsidehook}{}
781 \newcommand*{\Rightsidehookend}{}
782

```

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.

```

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

```

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 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.

```

804 \newcount\num@linesR
805 \newbox\one@lineR

```

806 \newcount\par@lineR

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

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

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

```
807
808 \newcounter{pstartL}
809 \newcounter{pstartLold}
810 \renewcommand{\thepstartL}{\bfseries\@arabic{c@pstartL}. }
811 \newcounter{pstartR}
812 \newcounter{pstartRold}
813 \renewcommand{\thepstartR}{\bfseries\@arabic{c@pstartR}. }
814
815 \newcommandx*{\pstartL}[1][1]{%
816   \if@nobreak%
817     \let\@oldnobreak\@nobreaktrue%
818   \else%
819     \let\@oldnobreak\@nobreakfalse%
820   \fi%
821   \@nobreaktrue%
822   \ifnumbering \else%
823     \led@err@PstartNotNumbered%
824     \beginnumbering%
825   \fi%
826   \ifnumberedpar@%
827     \led@err@PstartInPstart%
828   \pend%
829 \fi%
```

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

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

```

837 \global\advance\l@dnumpstartsL \@ne%
838 \ifnum\l@dnumpstartsL>\l@dc@maxchunks%
839   \led@err@TooManyPstarts%
840   \global\l@dnumpstartsL=\l@dc@maxchunks%
841 \fi%
842 \global\setnamebox{1@dLcolrawbox\the\l@dnumpstartsL}=\vbox\bgroup%
843 \ifautopar\else%
844   \ifnumberpstart%
845     \ifsidepstartnum%
846     \else%
847       \thepstartL%
848     \fi%
849   \fi%
850 \fi%
851 \hsize=Lcolwidth%
852 \numberedpar@true%
853 \iflabelpstart\protected@edef\@currentlabel%
854   {\p@pstartL\thepstartL}\fi%
855
856 Dump the optional arguments
855 \ifstrempy{#1}%
856   {}%
857   {\csgdef{before@pstartL@the\l@dnumpstartsL}{\noindent#1}}%
858 }
859 \newcommandx*{\pstartR}[1][1]{%
860   \if@nbreak%
861     \let\@oldnbreak\@nbreaktrue%
862   \else%
863     \let\@oldnbreak\@nbreakfalse%
864   \fi%
865   \@nbreaktrue%
866   \ifnumberingR \else%
867     \led@err@PstartNotNumbered%
868     \beginnumberingR%
869   \fi%
870   \ifnumberedpar@%
871     \led@err@PstartInPstart%
872     \pendR%
873   \fi%
874   \ifpst@rtedR\else%
875     \setcounter{pstartRold}{\value{pstartR}}%
876     \list@clear{\inserts@listR}%
877     \global\let\next@insertR=\empty%
878     \global\pst@rtedRtrue%
879   \fi%
880   \begingroup\normal@pars%
881   \global\advance\l@dnumpstartsR \@ne%
882   \ifnum\l@dnumpstartsR>\l@dc@maxchunks%
883     \led@err@TooManyPstarts%
884     \global\l@dnumpstartsR=\l@dc@maxchunks%

```



```

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

```

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

```

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

```

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

```

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

```

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

```

Dump content of the optional argument.

```

919 \ifstrempy{#1}%
920   {}%
921   {\csgdef{after@pendL@the\l@dnumpstartsL}{\noindent#1}}%
922 }

```

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

```

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

17.2 Processing one line

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

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

```

945 \newbox\l@dleftbox
946 \newbox\l@drightbox
947
```

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

```

\countRline 948 \newcount\countLline
949   \countLline \z@
950 \newcount\countRline
951   \countRline \z@
952
```

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

```

\@donetotallinesL 953 \newcount\@donereallinesL
954 \newcount\@donetotallinesL
\@donereallinesR 955 \newcount\@donereallinesR
```

```
956 \newcount\@donetotallinesR
957
```

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

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

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

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

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

```

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

```

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

```

\doinlinedlineLhook 1019 \newcommand*{\dolineLhook}[1]{\gdef\dolineLhook{#1}}%
\doinlinedlineRhook 1020 \newcommand*{\dolineRhook}[1]{\gdef\dolineRhook{#1}}%
1021 \newcommand*{\doinlinedlineLhook}[1]{\gdef\doinlinedlineLhook{#1}}%
1022 \newcommand*{\doinlinedlineRhook}[1]{\gdef\doinlinedlineRhook{#1}}%
1023

```

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

```

\dolineRhook 1024 \newcommand*{\do@lineLhook}{}
\doinlinedlineLhook 1025 \newcommand*{\do@lineRhook}{}
\doinlinedlineRhook 1026 \newcommand*{\do@insidelineLhook}{}
1027 \newcommand*{\do@insidelineRhook}{}
1028

```

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

```

1029 \newcommand*{\do@lineR}{%
1030   \ledRcol@true%
1031   \advance\countRline \one
1032   \ifvbox\namebox{\l@dRcolrawbox\the\l@dpscR}%
1033   {\vbadness=10000
1034     \splittopskip=\z@
1035     \do@lineRhook
1036     \l@demptyd@ta
1037     \global\setbox\one@lineR=\vsplit\namebox{\l@dRcolrawbox\the\l@dpscR}

```

```

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

```

```

\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.

```

1065 \newcommand*{\getline@numR}{%
1066   \global\advance\absline@numR \@ne
1067   \do@actionsR
1068   \do@ballastR
1069   \ifledgroupnotesR\else\ifnumberline
1070     \ifsublines@
1071       \ifnum\sub@lockR<\tw@
1072         \global\advance\subline@numR \@ne
1073       \fi
1074     \else
1075       \ifnum\@lockR<\tw@
1076         \global\advance\line@numR \@ne
1077         \global\subline@numR \z@
1078       \fi
1079     \fi

```

```

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

```

`\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.

```

1103 \newcommand*{\do@ballastR}{\global\ballast@count=\z@
1104   \begingroup
1105     \advance\absline@numR \@ne
1106     \ifnum\next@actionlineR=\absline@numR
1107       \ifnum\next@actionR>-1001
1108         \global\advance\ballast@count by -\c@ballast
1109       \fi
1110     \fi
1111   \endgroup}

```

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

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

```

1112 \newcommand*{\do@actions@fixedcodeR}{%
1113   \ifcase\@l@temptcnta%
1114   \or% % 1001
1115     \global\sublines@true
1116   \or% % 1002
1117     \global\sublines@false
1118   \or% % 1003
1119     \global\@lockR=\@ne

```

```

1120 \or% % 1004
1121 \ifnum\@lockR=\tw@
1122 \global\@lockR=\thr@@
1123 \else
1124 \global\@lockR=\z@
1125 \fi
1126 \or% % 1005
1127 \global\sub@lockR=\@ne
1128 \or% % 1006
1129 \ifnum\sub@lockR=\tw@
1130 \global\sub@lockR=\thr@@
1131 \else
1132 \global\sub@lockR=\z@
1133 \fi
1134 \or% % 1007
1135 \l@dskipnumbertrue
1136 \else
1137 \led@warn@BadAction
1138 \fi}
1139
1140
1141 \newcommand*{\do@actionsR}{%
1142 \global\let\do@actions@nextR=\relax
1143 \@l@dttempcntb=\absline@numR
1144 \ifnum\@l@dttempcntb<\next@actionlineR\else
1145 \ifnum\next@actionR>-1001\relax
1146 \global\page@numR=\next@actionR
1147 \ifbypage@R
1148 \global\line@numR \z@ \global\subline@numR \z@
1149 \fi
1150 \else
1151 \ifnum\next@actionR<-4999\relax % 9/05 added relax here
1152 \@l@dttempcnta=-\next@actionR
1153 \advance\@l@dttempcnta by -5001\relax
1154 \ifsublines@
1155 \global\subline@numR=\@l@dttempcnta
1156 \else
1157 \global\line@numR=\@l@dttempcnta
1158 \fi
1159 \else
1160 \@l@dttempcnta=-\next@actionR
1161 \advance\@l@dttempcnta by -1000\relax
1162 \do@actions@fixedcodeR
1163 \fi
1164 \fi
1165 \ifx\actionlines@listR\empty
1166 \gdef\next@actionlineR{1000000}%
1167 \else
1168 \gl@p\actionlines@listR\to\next@actionlineR
1169 \gl@p\actions@listR\to\next@actionR

```

```

1170     \global\let\do@actions@nextR=\do@actionsR
1171     \fi
1172   \fi
1173   \do@actions@nextR}
1174

```

17.4 Line number printing

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

```

\ch@cksub@l@ckR 1175
\ch@ck@l@ckR 1176 \providecommand*\l@dcalcnum}[3]{%
\fx@l@cksR 1177   \ifnum #1 > #2\relax
\affixline@numR 1178     \@l@tempcnta = #1\relax
1179     \advance\@l@tempcnta by -#2\relax
1180     \divide\@l@tempcnta by #3\relax
1181     \multiply\@l@tempcnta by #3\relax
1182     \advance\@l@tempcnta by #2\relax
1183   \else
1184     \@l@tempcnta=#2\relax
1185   \fi}
1186
1187 \newcommand*\ch@cksub@l@ckR}{%
1188   \ifcase\sub@lockR
1189   \or
1190     \ifnum\sublock@disp=\@ne
1191       \@l@tempcntb \z@ \@l@tempcnta \@ne
1192     \fi
1193   \or
1194     \ifnum\sublock@disp=\tw@
1195     \else
1196       \@l@tempcntb \z@ \@l@tempcnta \@ne
1197     \fi
1198   \or
1199     \ifnum\sublock@disp=\z@
1200       \@l@tempcntb \z@ \@l@tempcnta \@ne
1201     \fi
1202   \fi}
1203
1204 \newcommand*\ch@ck@l@ckR}{%
1205   \ifcase\@lockR
1206   \or
1207     \ifnum\lock@disp=\@ne
1208       \@l@tempcntb \z@ \@l@tempcnta \@ne
1209     \fi
1210   \or
1211     \ifnum\lock@disp=\tw@
1212     \else
1213       \@l@tempcntb \z@ \@l@tempcnta \@ne
1214     \fi

```



```

1215 \or
1216 \ifnum\lock@disp=\z@
1217 \l@tempcntb \z@ \l@tempcnta \@ne
1218 \fi
1219 \fi}
1220
1221 \newcommand*{\f@x@l@cksR}{%
1222 \ifcase\@lockR
1223 \or
1224 \global\@lockR \tw@
1225 \or \or
1226 \global\@lockR \z@
1227 \fi
1228 \ifcase\sub@lockR
1229 \or
1230 \global\sub@lockR \tw@
1231 \or \or
1232 \global\sub@lockR \z@
1233 \fi}
1234
1235
1236 \newcommand*{\affixline@numR}{%
1237 \ifledgroupnotesR\else\ifnumberline
1238 \ifl@dskipnumber
1239 \global\l@dskipnumberfalse
1240 \else
1241 \ifsublines@
1242 \l@tempcntb=\subline@numR
1243 \l@dcalcnm{\subline@numR}{\c@firstsublinenumR}{\c@sublinenumincrementR}%
1244 \ch@cksub@lockR
1245 \else
1246 \l@tempcntb=\line@numR
1247 \ifx\linenumberlist\empty
1248 \l@dcalcnm{\line@numR}{\c@firstlinenumR}{\c@linenumincrementR}%
1249 \else
1250 \l@tempcnta=\line@numR
1251 \edef\rem@inder{\linenumberlist,\number\line@numR,%
1252 \edef\sc@n@list{\def\noexpand\sc@n@list
1253 ###1,\number\l@tempcnta,###2|{\def\noexpand\rem@inder{###2}}}%
1254 \sc@n@list\expandafter\sc@n@list\rem@inder|
1255 \ifx\rem@inder\empty\advance\l@tempcnta\@ne\fi
1256 \fi
1257 \ch@ck@l@ckR
1258 \fi
1259 \ifnum\l@tempcnta=\l@tempcntb
1260 \iftwocolumn
1261 \if@firstcolumn
1262 \gdef\l@dld@ta{\llap{\leftlinenumR}}}%
1263 \else
1264 \gdef\l@drd@ta{\rlap{\rightlinenumR}}}%

```

```

1265 \fi
1266 \else
1267 \l@dttempcntb=\line@marginR
1268 \ifnum\l@dttempcntb>\@ne
1269 \advance\l@dttempcntb by\page@numR
1270 \fi
1271 \ifodd\l@dttempcntb
1272 \gdef\l@drd@ta{\rlap{\rightlinenumR}}}%
1273 \else
1274 \gdef\l@dld@ta{\llap{\leftlinenumR}}}%
1275 \fi
1276 \fi
1277 \fi
1278 \f@x@l@cksR
1279 \fi
1280 \fi
1281 \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 begining of this command.

```

\affixpstart@numL
\affixpstart@numR 1282
\leftpstartnumR 1283 \newcommand*{\affixpstart@numL}{%
\rightpstartnumR 1284 \ifsidepstartnum
\leftpstartnumL 1285 \if@twocolumn
\rightpstartnumL 1286 \if@firstcolumn
\ifpstartnumR 1287 \gdef\l@dld@ta{\llap{\leftpstartnumL}}}%
1288 \else
1289 \gdef\l@drd@ta{\rlap{\rightpstartnumL}}}%
1290 \fi
1291 \else
1292 \l@dttempcntb=\line@margin
1293 \ifnum\l@dttempcntb>\@ne
1294 \advance\l@dttempcntb \page@num
1295 \fi
1296 \ifodd\l@dttempcntb
1297 \gdef\l@drd@ta{\rlap{\rightpstartnumL}}}%
1298 \else
1299 \gdef\l@dld@ta{\llap{\leftpstartnumL}}}%
1300 \fi
1301 \fi
1302 \fi

```

```

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

```

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.

```
1348 \list@create{\inserts@listR}
```

`\add@insertsR` The right text version.

```
\add@inserts@nextR 1349 \newcommand*{\add@insertsR}{%
1350   \global\let\add@inserts@nextR=\relax
1351   \ifx\inserts@listR\empty \else
1352     \ifx\next@insertR\empty
1353       \ifx\insertlines@listR\empty
1354         \global\noteschanged@true
1355         \gdef\next@insertR{100000}%
1356       \else
1357         \gl@p\insertlines@listR\to\next@insertR
1358       \fi
1359     \fi
1360     \ifnum\next@insertR=\absline@numR
1361       \gl@p\inserts@listR\to\@insertR
1362       \@insertR
1363       \global\let\@insertR=\undefined
1364       \global\let\next@insertR=\empty
1365       \global\let\add@inserts@nextR=\add@insertsR
1366     \fi
1367   \fi
1368   \add@inserts@nextR}
1369
```

17.7 Penalties

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

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

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

```

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

```

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

```

1370 \newcommand*{\add@penaltiesL}{ }
1371 \newcommand*{\add@penaltiesR}{ }
1372

```

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.

```

1373 \newcommand*{\flush@notesR}{%
1374   \@xloop
1375   \ifx\inserts@listR\empty \else
1376     \gl@p\inserts@listR\to\@insertR
1377     \@insertR
1378     \global\let\@insertR=\undefined
1379   \repeat}
1380

```

18 Footnotes

18.1 Normal footnote formatting

The `\printlines` macro prints the line numbers for a note—which, in the general case, is a rather complicated task. The seven parameters of the argument are the line numbers as stored in `\l@d@nums`, in the form described on page ??: the

starting page, line, and sub-line numbers, followed by the ending page, line, and sub-line numbers, and then the font specifier for the lemma.

```

\printlinesR This is the right text version of \printlines and takes account of \Rlineflag.
\ledsavedprintlines Just in case, \ledsavedprintlines is a copy of the original \printlines.
    Just a reminder of the arguments:
\printlinesR #1 | #2 | #3 | #4 | #5 | #6 | #7
\printlinesR start-page | line | subline | end-page | line | subline | font
1381 \def\printlinesR#1|#2|#3|#4|#5|#6|#7|{\begingroup
1382 \setprintlines{#1}{#2}{#3}{#4}{#5}{#6}%
1383 \ifl@d@pnum #1\fullstop\fi
1384 \ifledplinenum \linenumr@p{#2}\Rlineflag\else \symp\linenum\fi
1385 \ifl@d@ssub \fullstop \sublinenumr@p{#3}\fi
1386 \ifl@d@dash \endashchar\fi
1387 \ifl@d@pnum #4\fullstop\fi
1388 \ifl@d@elin \linenumr@p{#5}\Rlineflag\fi
1389 \ifl@d@esl \ifl@d@elin \fullstop\fi \sublinenumr@p{#6}\fi
1390 \endgroup}
1391
1392 \let\ledsavedprintlines\printlines
1393

```

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.

```

1394 \list@create{\labelref@listR}
1395

```

`\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.

```

1396 \renewcommand*{\edlabel}[1]{\@bsphack
1397 \ifledRcol
1398 \write\linenum@outR{\string\@lab}%
1399 \ifx\labelref@listR\empty
1400 \xdef\label@refs{\zz@@@}%
1401 \else
1402 \glp\labelref@listR\to\label@refs
1403 \fi
1404 \ifvmode
1405 \advancelabel@refs
1406 \fi
1407 \protected@write\@auxout{%
1408 {\string\l@dmake@labelsR\space\thepage|\label@refs|\the\c@pstartR|{#1}}%

```

```

1409 \else
1410 \write\linenum@out{\string\@lab}%
1411 \ifx\labelref@list\empty
1412 \xdef\label@refs{\zz@@@}%
1413 \else
1414 \gl@p\labelref@list\to\label@refs
1415 \fi
1416 \ifvmode
1417 \advancelabel@refs
1418 \fi
1419 \protected@write\@auxout{%
1420 {\string\l@dmake@labels\space\thepage|\label@refs|\the\c@pstart|{#1}}%
1421 \fi
1422 \@esphack}
1423
1424

```

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

```

1425 \def\l@dmake@labelsR#1|#2|#3|#4|#5{%
1426 \expandafter\ifx\csname the@label#5\endcsname \relax\else
1427 \led@warn@DuplicateLabel{#4}%
1428 \fi
1429 \expandafter\gdef\csname the@label#5\endcsname{#1|#2\Rlineflag|#3|#4}%
1430 \ignorespaces}
1431 \AtBeginDocument{%
1432 \def\l@dmake@labelsR#1|#2|#3|#4|#5{%
1433 }
1434

```

`\@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.

```

1435 \renewcommand*{\@lab}{%
1436 \ifledRcol
1437 \xright@appenditem{\linenumr@p{\line@numR}}{%
1438 \ifsublines@ \sublinenumr@p{\subline@numR}\else 0\fi}%
1439 \to\labelref@listR
1440 \else
1441 \xright@appenditem{\linenumr@p{\line@num}}{%
1442 \ifsublines@ \sublinenumr@p{\subline@num}\else 0\fi}%
1443 \to\labelref@list
1444 \fi}
1445

```

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* 1446 \WithSuffix\newcommand\sidenotemargin*[1]{%
1447   \l@dgetsidenote@margin{#1}
1448   \global\sidenote@marginR=\@l@dttempcntb
1449   \global\sidenote@margin=\@l@dttempcntb
1450 }
1451 \newcount\sidenote@marginR
1452 \global\sidenote@margin=\@ne
1453
```

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

```
1454 \newcommand*\affixside@noteR{%
1455   \def\sidenotecontent@{%
1456     \numgdef{\itemcount@}{0}%
1457     \def\do##1{%
1458       \ifnumequal{\itemcount@}{0}%
1459         {%
1460           \appto\sidenotecontent@{##1}}% Not print not separator before the 1st note
1461           {\appto\sidenotecontent@{\sidenotesep ##1}%
1462           }%
1463           \numgdef{\itemcount@}{\itemcount@+1}%
1464         }%
1465       \dolistloop{\l@dcsnotetext}%
1466       \ifnumgreater{\itemcount@}{1}{\led@err@ManySidenotes}{}%
1467     \gdef\@templ@d{%
1468       \gdef\@templ@n{\l@dcsnotetext\l@dcsnotetext@l\l@dcsnotetext@r}%
1469       \ifx\@templ@d\@templ@n \else%
1470         \if@twocolumn%
1471           \if@firstcolumn%
1472             \setl@dlp@rbox{##1}{\sidenotecontent@}%
1473           \else%
1474             \setl@drp@rbox{\sidenotecontent@}%
1475           \fi%
1476         \else%
1477           \@l@dttempcntb=\sidenote@marginR%
1478           \ifnum\@l@dttempcntb>\@ne%
1479             \advance\@l@dttempcntb by\page@numR%
1480           \fi%
1481           \ifodd\@l@dttempcntb%
1482             \setl@drp@rbox{\sidenotecontent@}%
1483           \gdef\sidenotecontent@{%
1484             \numdef{\itemcount@}{0}%
1485             \dolistloop{\l@dcsnotetext@l}%
1486             \ifnumgreater{\itemcount@}{1}{\led@err@ManyLeftnotes}{}%

```



```

1487     \setl@dlp@rbox{\sidenotecontent@}%
1488   \else%
1489     \setl@dlp@rbox{\sidenotecontent@}%
1490     \gdef\sidenotecontent@{}%
1491     \numdef\itemcount@{0}%
1492     \dolistloop{\l@dcstotetext@r}%
1493     \ifnumgreater{\itemcount@}{1}{\led@err@ManyRightnotes}{}%
1494     \setl@drp@rbox{\sidenotecontent@}%
1495   \fi%
1496 \fi%
1497 \fi%
1498 }
1499

```

21 Familiar footnotes

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

```

1500 \renewcommand{\l@dbfnote}[1]{%
1501   \ifnumberedpar@
1502   \gdef\@tag{#1}%
1503   \ifledRcol%
1504     \xright@appenditem{\noexpand\v1@dbfnote{\csexpandonce{\@tag}}{\@thefnmark}}%
1505                       \to\inserts@listR
1506     \global\advance\insert@countR \@ne%
1507   \else%
1508     \xright@appenditem{\noexpand\v1@dbfnote{\csexpandonce{\@tag}}{\@thefnmark}}%
1509                       \to\inserts@list
1510     \global\advance\insert@count \@ne%
1511   \fi
1512 \fi\ignorespaces}
1513

```

`\normalbfnoteX`

```

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

```

```

1528 \ifluatex
1529 \footnotelang@lua%
1530 \fi
1531 \@ifundefined{xpg@main@language}%if polyglossia
1532 {}%
1533 {\footnotelang@poly}%
1534 \protected@csxdef{thisfootnote}{\csuse{thefootnote#1}}%
1535 \xright@appenditem{\noexpand\vbfnoteX{#1}{#2}{\csexpandonce{thisfootnote}}}%
1536 \to\inserts@list
1537 \global\advance\insert@count \@ne%
1538 \fi
1539 \fi\ignorespaces}
1540

```

22 Verse

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

```

\inserthangingsymbolL
\inserthangingsymbolR 1541 \newif\ifinserthangingsymbolR
1542 \newcommand{\inserthangingsymbolL}{%
1543 \ifinserthangingsymbol%
1544 \ifinstanzaL%
1545 \hangingsymbol%
1546 \fi%
1547 \fi}
1548 \newcommand{\inserthangingsymbolR}{%
1549 \ifinserthangingsymbolR%
1550 \ifinstanzaR%
1551 \hangingsymbol%
1552 \fi%
1553 \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 1554 \newcommand{\correcthangingL}{%
1555 \ifl@dpaging\else%
1556 \ifinstanzaL%
1557 \ifinserthangingsymbol%
1558 \hskip \@ifundefined{sza@00}{0}{\expandafter%
1559 \noexpand\csname sza@00\endcsname}\stanzaindentbase%
1560 \fi%
1561 \fi%
1562 \fi}

```

```

1563
1564 \newcommand{\correcthangingR}{%
1565 \ifl@dpaging\else%
1566   \ifinstanzaR%
1567     \ifinserthangingsymbolR%
1568       \hskip \@ifundefined{sza@00}{0}{\expandafter%
1569         \noexpand\csname sza@00\endcsname}\stanzaindentbase%
1570     \fi%
1571   \fi%
1572 \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.

```

1573 \chardef\next=\catcode'\&
1574 \catcode'\&=\active
1575

```

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

```

1576 \newenvironment{astanza}{%
1577   \startstanzahook
1578   \catcode'\&\active
1579   \global\stanza@count\@ne\stanza@modulo\@ne
1580   \ifnum\usernamecount{sza@00}=\z@
1581     \let\stanza@hang\relax
1582     \let\endlock\relax
1583   \else
1584   %% \interlinepenalty\@M % this screws things up, but I don't know why
1585   \rightskip\z@ plus 1fil\relax
1586   \fi
1587   \ifnum\usernamecount{szp@00}=\z@
1588     \let\sza@penalty\relax
1589   \fi
1590   \def&{%
1591     \endlock\mbox{}}%
1592     \sza@penalty
1593     \global\advance\stanza@count\@ne
1594     \@astanza@line}%
1595   \def\&{%
1596     \endlock\mbox{}}
1597   \pend
1598   \endstanzaextra}%
1599   \pstart
1600   \@astanza@line
1601 }{}
1602

```

`\@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.

```

1603 \newcommand*{\@astanza@line}{%
1604   \ifnum\value{stanzaindentsrepetition}=0
1605     \parindent=\csname sza@\number\stanza@count
1606             @\endcsname\stanzaindentbase
1607   \else
1608     \parindent=\csname sza@\number\stanza@modulo
1609             @\endcsname\stanzaindentbase
1610     \managestanza@modulo
1611   \fi
1612   \par
1613   \stanza@hang%\mbox{}%
1614   \ignorespaces}
1615

```

Lastly reset the modified category codes.

```

1616 \catcode'\&=\next
1617

```

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 1618 \providecommand*{\newnamebox}[1]{%
\unvnamebox 1619   \expandafter\newbox\csname #1\endcsname}
\namebox 1620 \providecommand*{\setnamebox}[1]{%
1621   \expandafter\setbox\csname #1\endcsname}
1622 \providecommand*{\unhnamebox}[1]{%
1623   \expandafter\unhbox\csname #1\endcsname}
1624 \providecommand*{\unvnamebox}[1]{%
1625   \expandafter\unvbox\csname #1\endcsname}
1626 \providecommand*{\namebox}[1]{%
1627   \csname #1\endcsname}
1628

\newnamecount Macros for creating and using ‘named’ counts.
\usenamecount 1629 \providecommand*{\newnamecount}[1]{%
1630   \expandafter\newcount\csname #1\endcsname}
1631 \providecommand*{\usenamecount}[1]{%
1632   \csname #1\endcsname}
1633

```

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.

```
1634 \newcount\l@dc@maxchunks
1635 \newcommand{\maxchunks}[1]{\l@dc@maxchunks=#1}
1636 \maxchunks{5120}
1637
```

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

`\l@dnumpstartsR` 1638 \newcount\l@dnumpstartsR
 1639

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

`\l@pscR` 1640 \newcount\l@dpscL
 1641 \newcount\l@dpscR
 1642

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

```
1643 \newcommand*\l@dsetuprawboxes{%
1644 \l@l@tempcntb=\l@dc@maxchunks
1645 \loop\ifnum\l@l@tempcntb>\z@
1646 \newnamebox{\l@dLcolrawbox\the\l@l@tempcntb}
1647 \newnamebox{\l@dRcolrawbox\the\l@l@tempcntb}
1648 \advance\l@l@tempcntb \m@ne
1649 \repeat}
1650
```

`\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.

```
1651 \newcommand*\l@dsetupmaxlinecounts{%
1652 \l@l@tempcntb=\l@dc@maxchunks
1653 \loop\ifnum\l@l@tempcntb>\z@
1654 \newnamecount{\l@dmaxlinesinpar\the\l@l@tempcntb}
1655 \advance\l@l@tempcntb \m@ne
1656 \repeat}
1657 \newcommand*\l@dzeromaxlinecounts{%
1658 \begingroup
1659 \l@l@tempcntb=\l@dc@maxchunks
1660 \loop\ifnum\l@l@tempcntb>\z@
```

```

1661 \global\usernamecount{l@dmxlinesinpar\the\@l@dttempcntb}=\z@
1662 \advance\@l@dttempcntb \m@ne
1663 \repeat
1664 \endgroup}
1665

```

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

```

1666 \AtBeginDocument{%
1667 \l@dsetuprawboxes
1668 \l@dsetupmaxlinecounts
1669 \l@dzeromaxlinecounts
1670 \l@dnumpestartsL=\z@
1671 \l@dnumpestartsR=\z@
1672 \l@dpscL=\z@
1673 \l@dpscR=\z@}
1674

```

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 1675 \newif\ifl@dusedbabel
\l@dusedbabeltrue 1676 \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 \.

```

```

1677 \newcommand*{\l@dbbl@set@language}[1]{%
1678 \edef\language{#1}%
1679 \select@language{\language}%

```

```

1680 \if@filesw
1681   \protected@write\@auxout{}\string\select@language{\language}%
1682   \addtocontents{toc}{\string\select@language{\language}}%
1683   \addtocontents{lof}{\string\select@language{\language}}%
1684   \addtocontents{lot}{\string\select@language{\language}}%
1685 \fi}
1686

```

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 1687 \providecommand{\selectlanguage}[1]{%
1688 \newcommand*{\l@duselanguage}[1]{%
1689 \gdef\theledlanguageL{
1690 \gdef\theledlanguageR{
1691

```

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

```

1692 \AtBeginDocument{%
1693   \ifundefined{xpg@main@language}{%
1694     \ifundefined{bbl@main@language}{%

```

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

```

1695   \l@dusedbabelfalse
1696   \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.

```

1697   \l@dusedbabeltrue
1698   \let\l@doldselectlanguage\selectlanguage
1699   \let\l@doldbbl@set@language\bbl@set@language
1700   \let\bbl@set@language\l@dbbl@set@language
1701   \renewcommand{\selectlanguage}[1]{%
1702     \l@doldselectlanguage{#1}%
1703     \ifledRcol \gdef\theledlanguageR{#1}%
1704     \else      \gdef\theledlanguageL{#1}%
1705   \fi}

```

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

```

1706   \renewcommand*{\l@duselanguage}[1]{%
1707     \l@doldselectlanguage{#1}}

```

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

```

1708   \gdef\theledlanguageL{\bbl@main@language}%
1709   \gdef\theledlanguageR{\bbl@main@language}%

```

```

1710    }%
1711  }

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

    That's it.
1721  }}

```

```

    \if@pstarts \check@pstarts returns \@pstartstrue if there are any unprocessed chunks.
    \@pstartstrue 1722 \newif\if@pstarts
    \@pstartstrue 1723 \newcommand*{\check@pstarts}{%
    \check@pstarts 1724 \@pstartstrue
    1725 \ifnum\l@dnumpstartsL>\l@dpscL
    1726 \@pstartstrue
    1727 \else
    1728 \ifnum\l@dnumpstartsR>\l@dpscR
    1729 \@pstartstrue
    1730 \fi
    1731 \fi
    1732 }
    1733

```

```

    \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 1734 \newif\ifaraw@text
    1735 \araw@textfalse
    1736 \newcommand*{\checkraw@text}{%
    1737 \araw@textfalse
    1738 \ifvbox\namebox{\l@dLcolrawbox\the\l@dpscL}
    1739 \araw@texttrue
    1740 \else
    1741 \ifvbox\namebox{\l@dRcolrawbox\the\l@dpscR}
    1742 \araw@texttrue
    1743 \fi
    1744 \fi
    1745 }
    1746

```

```

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

```



```

1747 \newcommand*{\@writelinesinparL}{%
1748   \edef\next{%
1749     \write\linenum@out{\string\@pend[\the\@donereallinesL]}}%
1750   \next
1751   \global\@donereallinesL \z@}
1752 \newcommand*{\@writelinesinparR}{%
1753   \edef\next{%
1754     \write\linenum@outR{\string\@pendR[\the\@donereallinesR]}}%
1755   \next
1756   \global\@donereallinesR \z@}
1757

```

26 Parallel columns

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

```

\@eledsectionR 1758 \newsavebox{\@eledsectionL}%
1759 \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.

```

1760 \newcommand*{\Columns}{%
1761   \eledsection@correcting@skip=-\baselineskip% Correction for sections' titles
1762   \setcounter{pstartL}{\value{pstartLold}}
1763   \setcounter{pstartR}{\value{pstartRold}}
1764   \ifnum\l@dnumpstartsL=\l@dnumpstartsR\else
1765     \led@err@BadLeftRightPstarts{\the\l@dnumpstartsL}{\the\l@dnumpstartsR}%
1766   \fi

```

Start a group and zero counters, etc.

```

1767 \begingroup
1768   \l@dzeropenalties
1769   \endgraf\global\num@lines=\prevgraf
1770   \global\num@linesR=\prevgraf
1771   \global\par@line=\z@
1772   \global\par@lineR=\z@
1773   \global\l@dpscL=\z@
1774   \global\l@dpscR=\z@

```

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

```

1775   \check@pstarts
1776   \loop\if@pstarts
1777     \global\pstartnumtrue
1778     \global\pstartnumRtrue

```

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

```

1779   \global\advance\l@dpscL \@ne
1780   \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

```

1781      \checkraw@text
1782 {      \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.

1783      \l@duselanguage{\theledlanguageL}%
1784      \do@lineL
1785      \xifinlist{\the\l@dpscL}{\eled@sections@L}%
1786      {%
1787      \ifdefstring{\@eledsectmark}{L}%
1788      {\csuse{eled@sectmark@the\l@dpscL}%
1789      }{}}%
1790      \global\csundef{eled@sectmark@the\l@dpscL}%
1791      \savebox{\@eledsectionL}{\parbox[t]{}[t]{\Lcolwidth}{\vbox{} \print@eledse
1792      }%
1793      {}}%
1794      \l@duselanguage{\theledlanguageR}%
1795      \do@lineR
1796      \xifinlist{\the\l@dpscR}{\eled@sections@R}%
1797      {%
1798      \ifdefstring{\@eledsectmark}{R}%
1799      {\csuse{eled@sectmark@the\l@dpscR }%
1800      }{}}%
1801      \global\csundef{eled@sectmark@the\l@dpscR }%
1802      \savebox{\@eledsectionR}{\parbox[t]{}[t]{\Rcolwidth}{\vbox{} \print@eledse
1803      }%
1804      \hb@xt@ \hsize{%
1805      \ifdefstring{\columns@position}{L}{-}{\hfill }%
1806      \unhbox\l@dleftbox%
1807      \ifhbox\@eledsectionL%
1808      \usebox{\@eledsectionL}%
1809      \fi%
1810      \print@columnseparator%
1811      \unhbox\l@drightbox%
1812      \ifhbox\@eledsectionR%
1813      \usebox{\@eledsectionR}%
1814      \fi%
1815      \ifdefstring{\columns@position}{R}{-}{\hfill}%
1816      }%
1817      \checkraw@text
1818      \checkverseL
1819      \checkverseR
1820      \checkpb@columns
1821      \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.

```

1822 \@writelinesinparL
1823 \@writelinesinparR
1824 \check@pstarts
1825 \ifbypstart@
1826 \write\linenum@out{\string\@set[1]}
1827 \resetprevline@
1828 \fi
1829 \ifbypstart@R
1830 \write\linenum@outR{\string\@set[1]}
1831 \resetprevline@
1832 \fi
1833 \addtocounter{pstartL}{1}
1834 \addtocounter{pstartR}{1}
1835 \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.

```

1836 \flush@notes
1837 \flush@notesR
1838 \endgroup
1839 \global\l@dpscL=\z@
1840 \global\l@dpscR=\z@
1841 \global\l@dnumpstartsL=\z@
1842 \global\l@dnumpstartsR=\z@
1843 \ignorespaces
1844 \global\instanzaLfalse
1845 \global\instanzaRfalse}
1846

```

`\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).

```

1847 \def\print@columnseparator{%
1848 \ifdim\beforecolumnseparator<0pt%
1849 \hfill%
1850 \else%
1851 \hspace{\beforecolumnseparator}%
1852 \fi%
1853 \columnseparator%
1854 \ifdim\aftercolumnseparator<0pt%
1855 \hfill%
1856 \else%
1857 \hspace{\beforecolumnseparator}%
1858 \fi%
1859 }%
1860 %\end{macrocode}
1861 % \end{macro}
1862 % \begin{macro}{\checkpb@columns}
1863 % \cs{checkpb@columns} prevent or make pagebreaking in columns, depending of the use of \cs{ledpb} on

```

```

1864 % \begin{macrocode}
1865
1866 \newcommand{\checkpb@columns}{%
1867   \newif\if@pb
1868   \newif\if@nopb
1869   \IfStrEq{\led@pb@setting}{before}{
1870     \numdef{\next@absline}{\the\absline@num+1}%
1871     \numdef{\next@abslineR}{\the\absline@numR+1}%
1872     \xifinlistcs{\next@absline}{l@prev@pb}{\@pbtrue}{}%
1873     \xifinlistcs{\next@abslineR}{l@prev@pbR}{\@pbtrue}{%
1874     \xifinlistcs{\next@absline}{l@prev@nopb}{\@nopbtrue}{}%
1875     \xifinlistcs{\next@abslineR}{l@prev@nopbR}{\@nopbtrue}{%
1876   }{}
1877   \IfStrEq{\led@pb@setting}{after}{
1878     \xifinlistcs{\the\absline@num}{l@prev@pb}{\@pbtrue}{}%
1879     \xifinlistcs{\the\absline@numR}{l@prev@pbR}{\@pbtrue}{%
1880     \xifinlistcs{\the\absline@num}{l@prev@nopb}{\@nopbtrue}{}%
1881     \xifinlistcs{\the\absline@numR}{l@prev@nopbR}{\@nopbtrue}{%
1882   }{}
1883 \if@nopb\nopagebreak[4]\enlargethispage{\baselineskip}\fi
1884 \if@pb\pagebreak[4]\fi
1885 }

```

`\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).

```

1886 \newcommand*{\columnseparator}{%
1887   \smash{\rule[-0.2\baselineskip]{\columnrulewidth}{1.05\baselineskip}}}
1888 \newdimen\columnrulewidth
1889 \columnrulewidth=\z@
1890

```

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

```

1891 \newcommand*{\columnspostion}[1]{%
1892   \xdef\columns@position{#1}%
1893   }%
1894 \xdef\columns@position{R}%

```

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

```

1895 \newlength{\beforecolumnseparator}%
1896 \setlength{\beforecolumnseparator}{-2pt}%
1897
1898 \newlength{\aftercolumnseparator}%
1899 \setlength{\aftercolumnseparator}{-2pt}%
1900

```

```

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

setwidthliketwocolumns@R 1901 \newcommand{\setwidthliketwocolumns@L}{%
tpositionliketwocolumns@R 1902 % Temporary dimension, initially equal to the standard hsize, i.e. text width
epositionliketwocolumns@R 1903 % \begin{macrocode}
1904 \newdimen\temp%
1905 \temp=\hsize%

Hsize : Left + Right width
1906 \hsize=\Lcolwidth%
1907 \advance\hsize\Rcolwidth%

Now, calculating the remaining space
1908 \advance\temp-\hsize%

And multiply the hsize by 2/3 of this space
1909 \multiply\temp by 2%
1910 \divide\temp by 3%
1911 \advance\hsize\temp%
1912 }%
1913
1914 \newcommand{\setpositionliketwocolumns@L}{%
1915 \renewcommand{\ledrlfill}{\hfill}%
1916 }%
1917
1918 \newcommand{\setnotespositionliketwocolumns@L}{%
1919 }%
1920
1921

1922 \newcommand{\setwidthliketwocolumns@C}{%
1923 % Temporary dimension, initially equal to the standard hsize, i.e. text width

1924 \newdimen\temp%
1925 \temp=\hsize%
1926 % Hsize : Left + Right width

1927 \hsize=\Lcolwidth%
1928 \advance\hsize\Rcolwidth%
1929 % Now, calculating the remaining space
1930 \advance\temp-\hsize%

And multiply the hsize by 1/2 of this space
1931 \divide\temp by 2%
1932 \advance\hsize\temp%
1933 }%
1934
1935 \newcommand{\setpositionliketwocolumns@C}{%

```

```

1936 \doinsidelinehook{\hfill}%
1937 \renewcommand{\ledrlfill}{\hfill}%
1938 }%
1939
1940 \newcommand{\setnotespositionliketwocolumns@C}{%
1941   \newdimen\temp%
1942   \newdimen\tempa%
1943   \temp=\hsize%
1944   \tempa=\Lcolwidth%
1945   \advance\tempa\Rcolwidth%
1946   \advance\temp-\tempa%
1947   \divide\temp by 2%
1948   \leftskip=\temp%
1949   \rightskip=-\temp%
1950 }%
1951
1952 \newcommand{\setwidthliketwocolumns@R}{%
  Temporary dimension, initially equal to the standard hsize, i.e. text width
1953   \newdimen\temp%
1954   \temp=\hsize%
  Hsize : Left + Right width
1955   \hsize=\Lcolwidth%
1956   \advance\hsize\Rcolwidth%
  Now, calculating the remaining space
1957   \advance\temp-\hsize%
  And multiply the hsize by 2/3 of this space
1958   \multiply\temp by 2%
1959   \divide\temp by 3%
1960   \advance\hsize\temp%
1961 }%
1962
1963 \newcommand{\setpositionliketwocolumns@R}{%
1964   \doinsidelinehook{\hfill}%
1965 }%
1966
1967 \newcommand{\setnotespositionliketwocolumns@R}{%
1968   \newdimen\temp%
1969   \newdimen\tempa%
1970   \temp=\hsize%
1971   \tempa=\Lcolwidth%
1972   \advance\tempa\Rcolwidth%
1973   \advance\temp-\tempa%
1974   \divide\temp by 2%
1975   \leftskip=\temp%
1976   \rightskip=-\temp%
1977 }%
1978

```

This is considerably more complicated than parallel columns.

elinesL Counts for the number of lines on a left or right page, and the smaller of the
elinesR number of lines on a pair of facing pages.

```

\l@dminpagelines 1979 \newcount\numpagelinesL
                  1980 \newcount\numpagelinesR
                  1981 \newcount\l@dminpagelines
                  1982

```

```

1983 \newcommand*{\Pages}{%
1984   \eledsection@correcting@skip=-2\baselineskip% line correcting for section titles.
1985   \setcounter{pstartL}{\value{pstartLold}}
1986   \setcounter{pstartR}{\value{pstartRold}}
1987   \parledgroup@notespacing@set@correction
1988   \typeout{}
1989   \typeout{***** PAGES *****}
1990   \ifnum\l@dnumpstartsL=\l@dnumpstartsR\else
1991     \led@err@BadLeftRightPstarts{\the\l@dnumpstartsL}{\the\l@dnumpstartsR}%
1992   \fi

```

```

1993 \cleartol@evenpage
1994 \begingroup
1995   \l@dzeropenalties
1996   \endgraf\global\num@lines=\prevgraf
1997     \global\num@linesR=\prevgraf
1998   \global\par@line=\z@
1999   \global\par@lineR=\z@
2000   \global\l@dpscL=\z@
2001   \global\l@dpscR=\z@
2002   \writtenlinesLfalse
2003   \writtenlinesRfalse

```

2004	\check@pstarts
2005	\loop\if@pstarts

2006	\global\advance\l@dpscl \@ne
2007	\global\advance\l@dpscr \@ne

2008 \getlinesfromparlistL

```

2009      \getlinesfromparlistR
2010      \l@dcalc@maxoftwo{\@cs@linesinparL}{\@cs@linesinparR}%
2011          {\usenamecount{l@dmaxlinesinpar\the\l@dpscL}}}%
2012      \check@pstarts
2013      \repeat

```

Zero the counts again, ready for the next bit.

```

2014      \global\l@dpscL=\z@
2015      \global\l@dpscR=\z@

```

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

```

2016      \getlinesfrompagelistL
2017      \getlinesfrompagelistR
2018      \l@dcalc@minoftwo{\@cs@linesonpageL}{\@cs@linesonpageR}%
2019          {\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.

```

2020      \check@pstarts
2021      \if@pstarts

```

Increment the chunk counts to get the first pair.

```

2022      \global\advance\l@dpscL \@ne
2023      \global\advance\l@dpscR \@ne

```

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

```

2024      \global\@donereallinesL=\z@
2025      \global\@donetotallinesL=\z@
2026      \global\@donereallinesR=\z@
2027      \global\@donetotallinesR=\z@

```

Start a loop over the boxes (chunks).

```

2028      \checkraw@text
2029 %      \begingroup
2030 {      \loop\ifaraw@text

```

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

```

2031      \checkpageL
2032      \l@duselanguage{\theledlanguageL}%
2033 %%%      \begingroup
2034 {      \loop\ifl@dsamepage%

```

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

```

2035      \ifdefstring{\@eledsectnotoc}{L}{\ledsectnotoc}{L}
2036      \csuse{before@pstartL@the\l@dpscL}
2037      \global\csundef{before@pstartL@the\l@dpscL}
2038      \do@lineL
2039      \xifinlist{\the\l@dpscL}{\eled@sections@@}

```



```

2040      {\print@eledsectionL}%
2041      {}%
2042      \advance\numpagelinesL \@ne
2043      \ifshiftedpstarts
2044          \ifdim\ht\l@dleftbox>0pt\hb@xt@ \hsize{\ledstrutL\unhbox\l@dleftbox}\fi%
2045      \else%
2046          \parledgroup@correction@notespacing{L}
2047          \hb@xt@ \hsize{\ledstrutL\unhbox\l@dleftbox}%
2048      \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.

```

2049      \checkpageL%
2050      \get@nextboxL%
2051      \checkverseL
2052      \checkpbL
2053      \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.

```

2054      \ifl@dpagfull
2055          \@writelinesonpageL{\the\numpagelinesL}%
2056      \else
2057          \@writelinesonpageL{1000}%
2058      \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.

```

2059      \numpagelinesL \z@
2060      \parledgroup@correction@notespacing@init
2061      \clearl@dleftpage }%

```

Now do the same for the right text.

```

2062      \checkpageR
2063      \l@duselanguage{\theledlanguageR}%
2064      {
2065          \loop\ifl@dsamepage%
2066              \initnumbering@sectcountR
2067              \ifdefstring{\@eledsectnotoc}{R}{\ledsectnotoc}{}
2068              \csuse{before@pstartR@the\l@dpscR}
2069              \global\csundef{before@pstartR@the\l@dpscR}
2070              \do@lineR
2071              \xifinlist{\the\l@dpscR}{\eled@sectionsR@@}
2072              {\print@eledsectionR}%
2073              {}%
2074          \advance\numpagelinesR \@ne
2075          \ifshiftedpstarts
2076              \ifdim\ht\l@drighbox>0pt\hb@xt@ \hsize{\ledstrutR\unhbox\l@drighbox}\fi%
2077          \else%

```

```

2077             \parledgroup@correction@notespacing{R}
2078             \hb@xt@ \hsize{\ledstrutR\unhbox\l@drightbox}%
2079         \fi
2080         \checkpageR%
2081         \get@nextboxR%
2082         \checkverseR
2083         \checkpbr
2084         \repeat
2085         \ifl@dpagfull
2086             \@writelinesonpageR{\the\numpagelinesR}%
2087         \else
2088             \@writelinesonpageR{1000}%
2089         \fi
2090         \numpagelinesR=\z@
2091         \parledgroup@correction@notespacing@init

```

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

```

2092         \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.

```

2093         \checkraw@text
2094         \ifaraw@text
2095             \getlinesfrompagelistL
2096             \getlinesfrompagelistR
2097             \l@dcalc@minoftwo{\@cs@linesonpageL}{\@cs@linesonpageR}%
2098                                 {\l@dminpagelines}%
2099         \fi
2100         \repeat}

```

We have now output the text from all the chunks.

```

2101     \fi

```

Make sure that there are no inserts hanging around.

```

2102     \flush@notes
2103     \flush@notesR
2104 \endgroup

```

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

```

2105 \global\l@dpscL=\z@
2106 \global\l@dpscR=\z@
2107 \global\l@dnumpestartsL=\z@
2108 \global\l@dnumpestartsR=\z@
2109 \global\instanzaLfalse
2110 \global\instanzaRfalse
2111 \ignorespaces}
2112

```

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

`\ledstrutR`

```

2113 \newcommand*{\ledstrutL}{\strut}
2114 \newcommand*{\ledstrutR}{\strut}
2115

```

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

```

2116 \providecommand{\cleartoevenpage}[1][\@empty]{%
2117   \clearpage
2118   \ifodd\c@page\hbox{ }#1\clearpage\fi}
2119 \newcommand*{\cleartol@devenpage}{%
2120   \ifdim\pagetotal<\topskip% on an empty page
2121   \else
2122     \clearpage
2123   \fi
2124   \ifodd\c@page\hbox{ }\clearpage\fi}
2125 \newcommand*{\clearl@dleftpage}{%
2126   \clearpage
2127   \ifodd\c@page\else
2128     \led@err@LeftOnRightPage
2129     \hbox{ }%
2130     \cleardoublepage
2131   \fi}
2132 \newcommand*{\clearl@drightpage}{%
2133   \clearpage
2134   \ifodd\c@page
2135     \led@err@RightOnLeftPage
2136     \hbox{ }%
2137     \cleartoevenpage
2138   \fi}
2139

```

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

```

\cs@linesinparR 2140 \newcommand*{\getlinesfromparlistL}{%
2141   \ifx\linesinpar@listL\empty
2142     \gdef\cs@linesinparL{0}%
2143   \else
2144     \gl@p\linesinpar@listL\to\cs@linesinparL
2145   \fi}
2146 \newcommand*{\getlinesfromparlistR}{%
2147   \ifx\linesinpar@listR\empty
2148     \gdef\cs@linesinparR{0}%
2149   \else
2150     \gl@p\linesinpar@listR\to\cs@linesinparR
2151   \fi}

```

2152

```

\@cs@linesonpageR 2153 \newcommand*{\getlinesfrompagelistL}{%
2154   \ifx\linesonpage@listL\empty
2155     \gdef\@cs@linesonpageL{1000}%
2156   \else
2157     \gl@p\linesonpage@listL\to\@cs@linesonpageL
2158   \fi}
2159 \newcommand*{\getlinesfrompagelistR}{%
2160   \ifx\linesonpage@listR\empty
2161     \gdef\@cs@linesonpageR{1000}%
2162   \else
2163     \gl@p\linesonpage@listR\to\@cs@linesonpageR
2164   \fi}
2165

```

```

2166 \newcommand*{\@writelinesonpageL}[1]{%
2167   \edef\next{\write\linenum@out{\string\@lopL{#1}}}%
2168   \next}
2169 \newcommand*{\@writelinesonpageR}[1]{%
2170   \edef\next{\write\linenum@outR{\string\@lopR{#1}}}%
2171   \next}
2172

```

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

<code>\ifl@dsamepage</code>	<code>\checkpageL</code> tests if the space and lines already taken on the page by text and foot-
<code>\l@dsamepagetrue</code>	notes is less than the constraints. If so, then <code>\ifl@dpagefull</code> is set FALSE and
<code>\l@dsamepagefalse</code>	
<code>\ifl@dpagefull</code>	
<code>\l@dpagefulltrue</code>	
<code>\l@dpagefullfalse</code>	
<code>\checkpageL</code>	
<code>\checkpageR</code>	

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

```

2186 \newif\ifl@dsamepage
2187 \l@dsamepagetrue
2188 \newif\ifl@dpagfull
2189
2190 \newcommand*{\checkpageL}{%
2191   \l@dpagfulltrue
2192   \l@dsamepagetrue
2193   \check@goal
2194   \ifdim\pagetotal<\ledthegoal
2195     \ifnum\numpagelinesL<\l@dminpagelines
2196       \else
2197         \l@dsamepagefalse
2198         \l@dpagfullfalse
2199       \fi
2200     \else
2201       \l@dsamepagefalse
2202       \l@dpagfulltrue
2203     \fi}
2204 \newcommand*{\checkpageR}{%
2205   \l@dpagfulltrue
2206   \l@dsamepagetrue
2207   \check@goal
2208   \ifdim\pagetotal<\ledthegoal
2209     \ifnum\numpagelinesR<\l@dminpagelines
2210       \else
2211         \l@dsamepagefalse
2212         \l@dpagfullfalse
2213       \fi
2214     \else
2215       \l@dsamepagefalse
2216       \l@dpagfulltrue
2217     \fi}
2218

```

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

```

2219 \newcommand{\checkpbL}{
2220   \IfStrEq{\led@pb@setting}{after}{
2221     \xifinlistcs{\the\absline@num}{\l@prev@pb}{\l@dpagfulltrue\l@dsamepagefalse}{
2222     \xifinlistcs{\the\absline@num}{\l@prev@nopb}{\l@dpagfullfalse\l@dsamepagetrue}{
2223   }}
2224   \IfStrEq{\led@pb@setting}{before}{
2225     \numdef{\next@absline}{\the\absline@num+1}
2226     \xifinlistcs{\next@absline}{\l@prev@pb}{\l@dpagfulltrue\l@dsamepagefalse}{

```

```

2227 \xifinlistcs{\next@absline}{l@prev@nopb}{\l@dpagfullfalse\l@dsamepagetrue}{\}
2228 }{\}
2229 }
2230
2231 \newcommand{\checkpbR}{
2232 \IfStrEq{\led@pb@setting}{after}{
2233 \xifinlistcs{\the\absline@numR}{l@prev@pbR}{\l@dpagfulltrue\l@dsamepagefalse}{\}
2234 \xifinlistcs{\the\absline@numR}{l@prev@nopbR}{\l@dpagfullfalse\l@dsamepagetrue}{\}
2235 }{\}
2236 \IfStrEq{\led@pb@setting}{before}{
2237 \numdef{\next@abslineR}{\the\absline@numR+1}
2238 \xifinlistcs{\next@abslineR}{l@prev@pbR}{\l@dpagfulltrue\l@dsamepagefalse}{\}
2239 \xifinlistcs{\next@abslineR}{l@prev@nopbR}{\l@dpagfullfalse\l@dsamepagetrue}{\}
2240 }{\}
2241 }

```

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

```

2242 \newcommand{\checkverseL}{
2243 \ifinstanzaL
2244 \iflednopbinverse
2245 \ifinserthangingsymbol
2246 \numgdef{\prev@abslineverse}{\the\absline@num-1}
2247 \IfStrEq{\led@pb@setting}{after}{\lednopbnum{\prev@abslineverse}}{\}
2248 \IfStrEq{\led@pb@setting}{before}{\ifnum\numpagelinesL<3\ledpbnum{\prev@abslineverse}}{\}
2249 \fi
2250 \fi
2251 \fi
2252 }
2253 \newcommand{\checkverseR}{
2254 \ifinstanzaR
2255 \iflednopbinverse
2256 \ifinserthangingsymbolR
2257 \numgdef{\prev@abslineverse}{\the\absline@numR-1}
2258 \IfStrEq{\led@pb@setting}{after}{\lednopbnumR{\prev@abslineverse}}{\}
2259 \IfStrEq{\led@pb@setting}{before}{\ifnum\numpagelinesR<3\ledpbnumR{\prev@abslineverse}}{\}
2260 \fi
2261 \fi
2262 \fi
2263 }

```

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

```

2264 \newdimen\ledthegoal
2265 \ifshiftedpstarts
2266 \newcommand*{\goalfraction}{0.95}
2267 \else
2268 \newcommand*{\goalfraction}{0.9}

```

```

2269 \fi
2270
2271 \newcommand*{\check@goal}{%
2272   \ledthegoal=\goalfraction\pagegoal}
2273

```

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

```

\ifwritelinesL 2274 \newif\ifwritelinesL
                2275 \newif\ifwritelinesR
                2276

```

`\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.

```

2277 \newcommand*{\get@nextboxL}{%
2278   \ifvbox\namebox{\l@dLcolrawbox\the\l@dpscl}% box is not empty

```

The current box is not empty; do nothing.

```

2279   \else%                                     box is empty

```

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

```

2280   \ifnum\usenamecount{\l@dmaxlinesinpar\the\l@dpscl}>\@donetotallinesL
2281     \parledgroup@notes@endL
2282   \else

```

Sufficient lines have been output.

```

2283     \ifnum\usenamecount{\l@dmaxlinesinpar\the\l@dpscl}=\@donetotallinesL
2284       \parledgroup@notes@endL
2285     \fi
2286   \ifwritelinesL\else

```

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

```

2287     \@writelinesinparL
2288     \writelinesLtrue
2289   \fi
2290   \ifnum\l@dnumpstartsL>\l@dpscl

```

There are still unprocessed boxes. Recalculate the maximum number of lines needed, and move onto the next box (by incrementing `\l@dpscl`). If needed, restart the line numbering. Increment the `pstartL` counter.

```

2291     \writelinesLfalse
2292     \ifbypstart@
2293       \ifnum\value{pstartL}<\value{pstartLold}
2294       \else
2295         \global\line@num=0
2296         \resetprevline@
2297       \fi
2298     \fi

```

2299 % Add the content of the optional argument of the previous `\cs{pend}`.

```

2300 %   \begin{macrocode}
2301   \csuse{after@pendL@the\l@dpscl}%
2302   \global\csundef{after@pendL@the\l@dpscl}%

```

```

2303 %      \end{macrocode}
2304 %      \begin{macrocode}
2305          \addtocounter{pstartL}{1}
2306          \global\pstartnumtrue
2307          \l@dcalc@maxoftwo{\the\usernamecount{l@dmxlinesinpar\the\l@dpscL}}%
2308                      {\the\@donetotallinesL}%
2309                      {\usernamecount{l@dmxlinesinpar\the\l@dpscL}}%
2310          \global\@donetotallinesL \z@
2311          \global\advance\l@dpscL \@ne

Add notes of parallel ledgroup.

2312          \parledgroup@notes@endL
2313          \parledgroup@correction@notes@spacing@final{L}
2314      \else
2315 % Add the content of the optional argument of the last \cs{pend}.
2316 %      \begin{macrocode}
2317          \l@dpagetrue
2318          \l@dsamepagefalse%
2319          \csuse{after@pendL@the\l@dpscL}%
2320          \global\csundef{after@pendL@the\l@dpscL}%
2321 %      \end{macrocode}
2322 %      \begin{macrocode}
2323      \fi
2324  \fi
2325 \fi}

2326 \newcommand*{\get@nextboxR}{%
2327 \ifvbox\namebox{l@dRcolrawbox\the\l@dpscR}% box is not empty
2328 \else% box is empty
2329 \ifnum\usernamecount{l@dmxlinesinpar\the\l@dpscR}>\@donetotallinesR
2330 \parledgroup@notes@endR
2331 \else
2332 \ifnum\usernamecount{l@dmxlinesinpar\the\l@dpscR}=\@donetotallinesR
2333 \parledgroup@notes@endR
2334 \fi
2335 \ifwrittenlinesR\else
2336 \@writelinesinparR
2337 \writtenlinesRtrue
2338 \fi
2339 \ifnum\l@dnumpstartsR>\l@dpscR
2340 \writtenlinesRfalse
2341 \ifbypstartR
2342 \ifnum\value{pstartR}<\value{pstartRold}
2343 \else
2344 \global\line@numR=0
2345 \resetprevline@
2346 \fi
2347 \fi
2348 \csuse{after@pendR@the\l@dpscR}%
2349 \global\csundef{after@pendR@the\l@dpscR}%
2350 \addtocounter{pstartR}{1}

```



```

2351      \global\pstartnumRtrue
2352      \l@dcalc@maxoftwo{\the\usernamecount{l@dmaxlinesinpar\the\l@dpscR}}%
2353              {\the\@donetotallinesR}%
2354              {\usernamecount{l@dmaxlinesinpar\the\l@dpscR}}}%
2355      \global\@donetotallinesR \z@
2356      \global\advance\l@dpscR \@ne
2357      \parledgroup@notes@endR
2358      \parledgroup@correction@notespadding@final{R}
2359      \else
2360      \l@dpagefulltrue%
2361      \l@dsamepagefalse%
2362      \csuse{after@pendR@\the\l@dpscR}%
2363      \global\csundef{after@pendR@\the\l@dpscR}%
2364      \fi
2365      \fi
2366      \fi}
2367

```

28 Sections' titles' commands

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

```

2368 \newcommand{\eledsectnotoc}[1]{\xdef\@eledsectnotoc{#1}}
2369 \eledsectnotoc{R}

```

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

```

2370 \newcommand{\eledsectmark}[1]{\xdef\@eledsectmark{#1}}
2371 \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`.

```

2372 \newskip\eledsection@correcting@skip

```

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

```

2373 \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).

```

2374 \def\l@prev@pbR{}
2375 \def\l@prev@nopbR{}

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

2376 \newcommand{\ledpbR}{\write\linenum@outR{\string\led@pbR}}
2377 \newcommand{\ledpbnumR}[1]{\write\linenum@outR{\string\led@pbnumR{#1}}}
2378 \newcommand{\lednopbR}{\write\linenum@outR{\string\led@nopbR}}
2379 \newcommand{\lednopbnumR}[1]{\write\linenum@outR{\string\led@nopbnumR{#1}}}

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

2380 \newcommand{\led@pbR}{\listcsxadd{\l@prev@pbR}{\the\absline@numR}}
2381 \newcommand{\led@pbnumR}[1]{\listcsxadd{\l@prev@pbR}{#1}}
2382 \newcommand{\led@nopbR}{\listcsxadd{\l@prev@nopbR}{\the\absline@numR}}
2383 \newcommand{\led@nopbnumR}[1]{\listcsxadd{\l@prev@nopbR}{#1}}

```

30 Parallel ledgroup

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

```

2384 \newmarks\parledgroup@
2385 \newmarks\parledgroup@series
2386 \newmarks\parledgroup@type

```

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

```

2387 \newcommand{\parledgroup@notes@startL}{%
2388   \ifnum\usenamecount{\l@dmxlinesinpar\the\l@dpscL}>0%
2389     \IfStrEq{\splitfirstmarks\parledgroup@type}{footnoteX}{\csuse{bhooknoteX@splitfirstma
2390     \IfStrEq{\splitfirstmarks\parledgroup@type}{Xfootnote}{\csuse{bhookXnote@splitfirstma
2391   \fi%
2392   \global\ledgroupnotesL@true%
2393   \insert@noterule@ledgroup{L}%
2394 }
2395 \newcommand{\parledgroup@notes@startR}{%
2396   \ifnum\usenamecount{\l@dmxlinesinpar\the\l@dpscR}>0%
2397     \IfStrEq{\splitfirstmarks\parledgroup@type}{footnoteX}{\csuse{bhooknoteX@splitfirstma
2398     \IfStrEq{\splitfirstmarks\parledgroup@type}{Xfootnote}{\csuse{bhookXnote@splitfirstma
2399   \fi%
2400   \global\ledgroupnotesR@true%

```

```

2401 \insert@noterule@ledgroup{R}%
2402 }

```

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

```

2403 \newcommand{\parledgroup@notes@endL}{%
2404   \global\ledgroupnotesL=false%
2405 }
2406 \newcommand{\parledgroup@notes@endR}{%
2407   \global\ledgroupnotesR=false%
2408 }

```

`\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`

```

2409 \newcommand{\insert@noterule@ledgroup}[1]{
2410   \IfStrEq{\splitbotmarks\parledgroup@}{begin}{%
2411     \IfStrEq{\splitbotmarks\parledgroup@type}{Xfootnote}{
2412       \csuse{ifledgroupnotes#1@}
2413       \vskip\skip\csuse{mp\splitbotmarks\parledgroup@series footins}
2414       \csuse{\splitbotmarks\parledgroup@series footnoterule}
2415       \fi
2416     }
2417   {}
2418   \IfStrEq{\splitbotmarks\parledgroup@type}{footnoteX}{
2419     \csuse{ifledgroupnotes#1@}
2420     \vskip\skip\csuse{mpfootins\splitbotmarks\parledgroup@series}
2421     \csuse{footnoterule\splitbotmarks\parledgroup@series}
2422     \fi
2423   }{}
2424 }
2425 {}
2426 }

```

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

```

2427 \newcommand{\parledgroupnotespacing}{}

```

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

```

2428 \dimdef{\parledgroup@notespacing@correction}{0pt}
2429 \newcommand{\parledgroup@notespacing@set@correction}{%
2430   {\notefontsetup\parledgroupnotespacing\dimgdef{\temp@spacing}{\baselineskip}}%
2431   \dimgdef{\parledgroup@notespacing@correction}{\baselineskip-\temp@spacing}%
2432 }

```

`\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.

```

2433 \newcommand{\parledgroup@correction@notespacing@init}{
2434   \dimdef{\parledgroup@notespacing@correction@accumulated}{0pt}
2435   \dimdef{\parledgroup@notespacing@correction@modulo}{0pt}
2436 }
2437 \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`.

```

2438 \newcommand{\parledgroup@correction@notespacing@final}[1]{
2439   \ifparledgroup
2440     \vspace{\parledgroup@notespacing@correction@accumulated}
2441     \parledgroup@correction@notespacing@init%
2442     \ifstrequal{#1}{L}{
2443       \numdef{\@checking}{\the\l@dpscL-1}
2444     }{
2445       \numdef{\@checking}{\the\l@dpscR-1}
2446     }
2447     \dimdef{\@beforenotes@current@diff}{\csuse{\parledgroup@beforenotes@\@checking L}-\cs
2448     \ifstrequal{#1}{L}%
2449     {% Left
2450       \ifdingreater{\@beforenotes@current@diff}{0pt}{\vspace{-\@beforenotes@current@diff}
2451     }%
2452     {% Right
2453       \ifdingreater{\@beforenotes@current@diff}{0pt}{\vspace{\@beforenotes@current@diff}}
2454     }%
2455   \fi
2456 }

```

`\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.

```

2457 {}
2458 \newcommand{\parledgroup@correction@notespacing}[1]{%
2459   \csuse{ifledgroupnotes#1@}%
2460   \vspace{-\parledgroup@notespacing@correction}%
2461   \dimdef{\parledgroup@notespacing@correction@accumulated}{\parledgroup@notespacing@correction@accumulated}

```

```

2462     \dimdef{\parledgroup@notespacing@correction@modulo}{\parledgroup@notespacing@correction@modulo+
2463     \ifdimless{\parledgroup@notespacing@correction@modulo}{\baselineskip}{\advance\numpagelinesL
2464     \dimdef{\parledgroup@notespacing@correction@modulo}{\parledgroup@notespacing@correction@modulo-
2465     }% mean greater than equal
2466     \fi%
2467 }

```

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

```

2468 \dimdef\parledgroup@beforenotesL{0pt}
2469 \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.

```

2470 \newcommand{\parledgroup@beforenotes@save}[1]{
2471   \ifparledgroup
2472     \csdingdef{@parledgroup@beforenotes@the\csuse{1@dnumstarts#1}#1}{\csuse{parledgroup@beforenotes
2473     \csdingdef{parledgroup@beforenotes#1}{0pt}
2474   \fi
2475 }

```

31 The End

i/codei

Appendix A Some things to do when changing version

Appendix A.1 Migration to eledpar 1.4.3

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

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

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

See the two following examples:

With standard `\hangingsymbol`:

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

With the modification of `\hangingsymbol`:

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

References

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

Index

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

Symbols	<code>\@adv</code>	366, 631, 632
<code>\&</code>	1573, 1574, 1578, 1595, 1616	<code>\@afterindentfalse</code> 761
<code>\@M</code>	1584	<code>\@arabic</code> 222, 223, 810, 813

<code>\@astanza@line</code>	1594, 1600, <u>1603</u>	<code>\@nobreakefalse</code>	819, 863, 1004
<code>\@auxout</code>	1407, 1419, 1681	<code>\@nobreaktrue</code>	817, 821, 861, 865, 1004
<code>\@beforenotes@current@diff</code>		<code>\@nopbtrue</code>	1874, 1875, 1880, 1881
	2447, 2450, 2453	<code>\@oldnobreake</code>	817, 819, 861, 863, 914, 935
<code>\@chapter</code>	762	<code>\@pbtrue</code>	1872, 1873, 1878, 1879
<code>\@checking</code>	2443, 2445, 2447	<code>\@pend</code>	<u>573</u> , 1749
<code>\@cs@linesinparL</code>	2010, <u>2140</u>	<code>\@pendR</code>	<u>573</u> , 1754
<code>\@cs@linesinparR</code>	2010, <u>2140</u>	<code>\@pstartsfalse</code>	<u>1722</u>
<code>\@cs@linesonpageL</code>	2018, 2097, <u>2153</u>	<code>\@pstartstrue</code>	<u>1722</u>
<code>\@cs@linesonpageR</code>	2018, 2097, <u>2153</u>	<code>\@ref</code>	<u>545</u>
<code>\@currentlabel</code>	853, 896	<code>\@ref@reg</code>	571
<code>\@donereallinesL</code>		<code>\@chapter</code>	762
	<u>953</u> , 982, 1749, 1751, 2024	<code>\@set</code>	<u>398</u> , 638, 639, 1826, 1830
<code>\@donereallinesR</code>		<code>\@startstanza</code>	774, 775, 797, 798
	<u>953</u> , 1055, 1754, 1756, 2026	<code>\@tag</code>	668, 691, 1502
<code>\@donetotallinesL</code>	<u>953</u> , 983,	<code>\@templ@d</code>	1467, 1469
	986, 2025, 2280, 2283, 2308, 2310	<code>\@templ@n</code>	1468, 1469
<code>\@donetotallinesR</code>	<u>953</u> , 1056,	<code>\@writelinesinparL</code>	<u>1747</u> , 1822, 2287
	1059, 2027, 2329, 2332, 2353, 2355	<code>\@writelinesinparR</code>	<u>1747</u> , 1823, 2336
<code>\@eled@sectioningfalse</code>	1007	<code>\@writelinesonpageL</code>	2055, 2057, <u>2166</u>
<code>\@eled@sectioningtrue</code>	1005	<code>\@writelinesonpageR</code>	2086, 2088, <u>2166</u>
<code>\@eledsectionL</code>	<u>1758</u> , 1791, 1807, 1808	<code>\@xloop</code>	1374
<code>\@eledsectionR</code>	<u>1758</u> , 1802, 1812, 1813		
<code>\@eledsectmark</code>	1002, 1787, 1798, 2370		
<code>\@eledsectnotoc</code>	1001, 2035, 2066, 2368		
<code>\@insertR</code>	1361–1363, 1376–1378		
<code>\@l@dttempcnta</code>	439,		
	441, 443, 444, 448, 450, 452,		
	453, 1113, 1152, 1153, 1155,		
	1157, 1160, 1161, 1178–1182,		
	1184, 1191, 1196, 1200, 1208,		
	1213, 1217, 1250, 1253, 1255, 1259		
<code>\@l@dttempcntb</code>	178, 180, 182,		
	1143, 1144, 1191, 1196, 1200,		
	1208, 1213, 1217, 1242, 1246,		
	1259, 1267–1269, 1271, 1292–		
	1294, 1296, 1313–1315, 1317,		
	1448, 1449, 1477–1479, 1481,		
	1644–1648, 1652–1655, 1659–1662		
<code>\@lab</code>	558, 1398, 1410, <u>1435</u>		
<code>\@lock</code>	970, 1093		
<code>\@lockR</code>	56, 310, 312, 314, 327, 473,		
	489, 490, 492, 493, 521, 522,		
	524, 1042, 1075, 1119, 1121,		
	1122, 1124, 1205, 1222, 1224, 1226		
<code>\@lopL</code>	<u>582</u> , 2167		
<code>\@lopR</code>	<u>582</u> , 2170		
<code>\@nl</code>	<u>288</u> , 614, 616		
<code>\@nl@reg</code>	337		
<code>\@nl@regR</code>	288		

- \advanceline 630, 661
 - \affixline@num 977
 - \affixline@numR 1049, 1175
 - \affixpstart@numL 991, 1282
 - \affixpstart@numR 1282
 - \affixside@note 993, 999
 - \affixside@noteR 1454
 - \aftercolumnseparator . . 4, 1854, 1895
 - \appto 1460, 1461
 - \araw@textfalse 1734
 - \araw@texttrue 1734
 - astanza (environment) 8, 1576
 - \at@begin@pairs 738, 743, 744
 - \AtBeginDocument . . . 1431, 1666, 1692
 - \AtBeginPairs 4, 743
- B**
- \ballast@count 1103, 1108
 - \bbl@main@language 1708, 1709
 - \bbl@set@language 1699, 1700
 - \beforecolumnseparator
..... 4, 1848, 1851, 1857, 1895
 - \beginnumbering 7, 785, 824
 - \beginnumberingR . . . 42, 123, 785, 868
 - \bfseries 810, 813
 - \bypage@Rfalse 143, 158, 163
 - \bypage@Rtrue 143, 153
 - \bypstart@Rfalse 143, 154, 164
 - \bypstart@Rtrue 143, 159
- C**
- \c@ballast 1108
 - \c@chapter 104
 - \c@chapterR 104
 - \c@firstlinenumR 187, 1248
 - \c@firstsublinenumR 191, 1243
 - \c@linenumincrementR 187, 1248
 - \c@page 614, 616, 2118, 2124, 2127, 2134
 - \c@pstart 1420
 - \c@pstartL 810
 - \c@pstartR 813, 1408
 - \c@section 105
 - \c@sectionR 105
 - \c@sublinenumincrementR . . 191, 1243
 - \c@subsection 106
 - \c@subsectionR 106
 - \c@subsubsection 107
 - \c@subsubsectionR 107
 - \ch@ck@l@ckR 1175
 - \ch@cksub@l@ckR 1175
 - \ch@cksub@lockR 1244
 - \chapter 747, 748, 757
 - \chapterinpages 734, 748, 759
 - \chardef 1573
 - \check@goal 2193, 2207, 2264
 - \check@pstarts
..... 1722, 1775, 1824, 2004, 2012, 2020
 - \checkpageL 2031, 2049, 2186
 - \checkpageR 2062, 2080, 2186
 - \checkpb@columns . . . 1820, 1862, 1866
 - \checkpbl 2052, 2219
 - \checkpbr 2083, 2219
 - \checkkraw@text
..... 1734, 1781, 1817, 2028, 2093
 - \checkverseL 1818, 2051, 2242
 - \checkverseR 1819, 2082, 2242
 - \cleardoublepage 2130
 - \clearl@dleftpage 2061, 2116
 - \clearl@drighpage 2092, 2116
 - \cleartoevenpage 2116
 - \cleartol@devenpage 1993, 2116
 - \closeout 95, 595, 601, 604, 608
 - \columnrulewidth 4, 1886
 - \Columns 3, 1760
 - \columns@position . . 1805, 1815, 1891
 - \columnseparator 4, 1853, 1886
 - \columnspostion 4, 1891
 - \correcthangingL 995, 1554
 - \correcthangingR 1554
 - \countLline 948, 959
 - \countRline 948, 1031
 - \critext 666
 - \cs 1863, 2299, 2315
 - \csdimgdef 2472, 2473
 - \csexpandonce . . 1504, 1508, 1524, 1535
 - \csgdef 857, 900, 921, 942
 - \csundef 1008, 1790, 1801,
2037, 2068, 2302, 2320, 2349, 2363
 - \csuse 1006,
1523, 1534, 1788, 1799, 2036,
2067, 2301, 2319, 2348, 2362,
2389, 2390, 2397, 2398, 2412–
2414, 2419–2421, 2447, 2459, 2472
- D**
- \DeclareOption 8–11
 - \def@tempb 161
 - \dimdef 2428, 2434, 2435,
2447, 2461, 2462, 2464, 2468, 2469
 - \dimen 617, 618, 622–624, 628

- `\dingdef` 2430, 2431
`\divide` 1180, 1910, 1931, 1947, 1959, 1974
`\do@actions` 1085
`\do@actions@fixedcodeR` 1112
`\do@actions@nextR` 1112
`\do@actionsR` 1067, 1112
`\do@ballast` 1086
`\do@ballastR` 1068, 1103
`\do@insidelineLhook` .. 995, 1021, 1024
`\do@insidelineRhook` 1022, 1024
`\do@lineL` 958, 1784, 2038
`\do@lineLhook` 963, 1019, 1024
`\do@lineR` 1029, 1795, 2069
`\do@lineRhook` 1020, 1024, 1035
`\do@lockoff` 508
`\do@lockoffL` 532
`\do@lockoffR` 508
`\do@lockon` 473
`\do@lockonL` 505
`\do@lockonR` 473
`\doinsidelinehook` 1936, 1964
`\doinsidelineLhook` 1019
`\doinsidelineRhook` 1019
`\dolineLhook` 1019
`\dolineRhook` 1019
`\dolistloop` 1465, 1485, 1492
`\dummy@ref` 554
- E**
- `\edfont@info` 717, 720, 726, 729
`\edlabel` 1396
`\edtext` 689
`\eled@sectioningR@out` .. 70, 95, 2373
`\eled@sections@@` 978, 1004, 1785, 2039
`\eled@sectionsR@@` 67, 1050, 1796, 2070
`\eledpar@error` 21, 30, 33, 35
`\eledsection@correcting@skip` ...
..... 1016, 1761, 1984, 2372
`\eledsectmark` 12, 2370
`\eledsectnotoc` 12, 2368
`\empty` .. 80, 83, 272, 280, 679, 702,
715, 724, 833, 877, 1165, 1247,
1255, 1351–1353, 1364, 1375,
1399, 1411, 2141, 2147, 2154, 2160
`\end@lemmas` 679, 680, 702, 703
`\endashchar` 1386
`\endgraf` 910, 931, 1769, 1996
`\endline@num` 561, 567
`\endlock` 650, 1582, 1591, 1596
`\endnumbering` 7, 73, 127, 786
`\endnumberingR` 45, 73, 110, 122, 135, 786
`\endpage@num` 560, 567
`\endstanzaextra` 1598
`\endsub` 617
`\endsubline@num` 562, 568
`\enlargethispage` 1883
environments:
 `astanza` 8, 1576
 `Leftside` 5, 766
 `pages` 5, 734
 `pairs` 3, 734
 `Rightside` 5, 783
`\extensionchars` 62, 116, 132, 140
- F**
- `\f@x@l@cksR` 1175
`\first@linenum@out@Rfalse` .. 590, 596
`\first@linenum@out@Rtrue` 590
`\firstlinenum` 6, 196
`\firstlinenum*` 6, 196
`\firstsublinenum` 6, 196
`\firstsublinenum*` 6, 196
`\fix@page` 333, 340
`\flag@end`
. 617, 675, 685, 686, 698, 708, 709
`\flag@start`
. 617, 674, 675, 686, 697, 698, 709
`\flush@notes` 1836, 2102
`\flush@notesR` 1373, 1837, 2103
`\footnotelang@lua` 1518, 1529
`\footnotelang@poly` 1522, 1533
`\fullstop` . 235, 1383, 1385, 1387, 1389
- G**
- `\get@linelistfile` 268
`\get@nextboxL` 2050, 2277
`\get@nextboxR` 2081, 2277
`\getline@numL` 969, 1083
`\getline@numR` 1041, 1065
`\getlinesfrompagelistL`
..... 2016, 2095, 2153
`\getlinesfrompagelistR`
..... 2017, 2096, 2153
`\getlinesfromparlistL` ... 2008, 2140
`\getlinesfromparlistR` ... 2009, 2140
`\gl@p` 275, 276,
283, 284, 680, 703, 719, 728,
1168, 1169, 1357, 1361, 1376,
1402, 1414, 2144, 2150, 2157, 2163
`\goalfraction` 5, 2264

H

\hangingsymbol 10, 1545, 1551
 \hb@xt@ 976, 985, 995, 1048,
 1058, 1804, 2044, 2047, 2075, 2078
 \hsize 851, 894, 1804, 1905–
 1908, 1911, 1925, 1927, 1928,
 1930, 1932, 1943, 1954–1957,
 1960, 1970, 2044, 2047, 2075, 2078

I

\if@filesw 1680
 \if@firstcolumn 1261, 1286, 1307, 1471
 \if@ledgroup 600
 \if@nobreak 816, 860
 \if@noeled@sec 68, 94
 \if@nopb 1868, 1883
 \if@pb 1867, 1884
 \if@pstarts 1722, 1776, 2005, 2021
 \if@RTL 675, 686, 698, 709, 1009
 \ifaraw@text ... 1734, 1782, 2030, 2094
 \ifautopar 843, 887
 \ifbypage@ 356
 \ifbypage@R 143, 346, 1147
 \ifbypstart@ 575, 1825, 2292
 \ifbypstart@R ... 143, 579, 1829, 2341
 \ifdefstring 1001, 1002,
 1787, 1798, 1805, 1815, 2035, 2066
 \ifdim ... 618, 622, 624, 628, 1848,
 1854, 2044, 2075, 2120, 2194, 2208
 \ifdimgreater 2450, 2453
 \ifdimless 2463
 \iffirst@linenum@out@R 590, 594
 \ifhbox 1807, 1812
 \ifinserthangingsymbol
 1543, 1557, 2245
 \ifinserthangingsymbolR
 1541, 1549, 1567, 2256
 \ifinstanzaL 764, 764, 1544, 1556, 2243
 \ifinstanzaR 764, 765, 1550, 1566, 2254
 \ifl@d@dash 1386
 \ifl@d@elin 1388, 1389
 \ifl@d@esl 1389
 \ifl@d@pnum 1383, 1387
 \ifl@d@ssub 1385
 \ifl@dpagefull 2054, 2085, 2186
 \ifl@dpadding 13, 1555, 1565
 \ifl@dpairing 13, 77
 \ifl@dsamelang 1677
 \ifl@dsamepage 2034, 2064, 2186
 \ifl@dskipnumber 1238

\ifl@dusedbabel 1675
 \iflabelpstart 853, 896
 \ifledgroupnotesL@ 1087
 \ifledgroupnotesR@ 1069, 1237
 \iflednopbinverse 2244, 2255
 \ifledplinenum 1384
 \ifledRcol 13, 179,
 201, 205, 209, 213, 255, 270,
 334, 343, 368, 382, 399, 416,
 428, 436, 457, 502, 529, 538,
 547, 619, 625, 631, 638, 646,
 651, 655, 660, 670, 693, 714,
 1397, 1436, 1503, 1516, 1703, 1715
 \ifluatex 1517, 1528
 \ifnoteschanged@ 87
 \ifnumberedpar@
 ... 826, 870, 906, 927, 1501, 1515
 \ifnumbering 148, 822, 903
 \ifnumberingR ... 43, 74, 112, 866, 924
 \ifnumberline 1069, 1087, 1237
 \ifnumberpstart ... 844, 888, 915, 936
 \ifnumequal 1458
 \ifnumgreater 1466, 1486, 1493
 \ifodd 1271, 1296,
 1317, 1481, 2118, 2124, 2127, 2134
 \ifparledgroup 2439, 2471
 \ifpst@rtedL 38, 830
 \ifpst@rtedR 38, 874
 \ifpstartnum 1327, 1332
 \ifpstartnumR 1282
 \ifshiftedpstarts 5, 2043, 2074, 2265
 \ifsidepstartnum 845, 889, 1284, 1305
 \ifstreempty 855, 898, 919, 940
 \IfStrEq 967, 1039, 1869,
 1877, 2220, 2224, 2232, 2236,
 2247, 2248, 2258, 2259, 2389,
 2390, 2397, 2398, 2410, 2411, 2418
 \ifstrequal 2442, 2448
 \ifsublines@ 233,
 322, 367, 400, 407, 438, 447,
 459, 466, 478, 512, 566, 568,
 1070, 1088, 1154, 1241, 1438, 1442
 \ifvbox 960, 1032, 1738, 1741, 2278, 2327
 \ifvnode 1404, 1416
 \ifwidthliketwocolumns 11
 \ifwrittenlinesL 2274, 2286
 \ifwrittenlinesR 2275, 2335
 \initnumbering@sectcmd 737, 751
 \initnumbering@sectcountR
 66, 99, 119, 2065

- \InputIfFileExists 69
 - \insert@count 544, 671, 694, 1510, 1537
 - \insert@countR 545, 670, 693, 1506, 1526
 - \insert@noterule@ledgroup
..... 2393, 2401, 2409
 - \inserthangingsymbolfalse 973
 - \inserthangingsymbolL 995, 1541
 - \inserthangingsymbolR 1541
 - \inserthangingsymbolRfalse 1045
 - \inserthangingsymbolRtrue 1043
 - \inserthangingsymboltrue 971
 - \insertlines@listR
.... 80, 243, 257, 550, 1353, 1357
 - \inserts@list 832, 1509, 1536
 - \inserts@listR 876, 1348,
1351, 1361, 1375, 1376, 1505, 1525
 - \instanzaLfalse 1844, 2109
 - \instanzaLtrue 775
 - \instanzaRfalse 1845, 2110
 - \instanzaRtrue 798
 - \interlinepenalty 1584
 - \itemcount@ 1456, 1458,
1463, 1466, 1484, 1486, 1491, 1493
- L**
- \l@d@nums 717, 720, 726, 729
 - \l@d@set 415, 646, 647
 - \l@d@bbl@set@language 1677, 1700
 - \l@d@bfnote 1500
 - \l@d@dc@maxchunks 838, 840,
882, 884, 1634, 1644, 1652, 1659
 - \l@d@dc@l@maxoftwo
..... 2010, 2173, 2307, 2352
 - \l@d@dc@l@minoftwo .. 2018, 2097, 2173
 - \l@d@dc@l@num 1175
 - \l@d@dc@checklang 1677
 - \l@d@dc@set@num 289, 292, 415
 - \l@d@dc@notetext 1465, 1468
 - \l@d@dc@notetext@l 1468, 1485
 - \l@d@dc@notetext@r 1468, 1492
 - \l@d@demptyd@ta 964, 1036
 - \l@d@dend@stuff 63, 117, 133, 141
 - \l@d@getline@margin 177
 - \l@d@getsidenote@margin 1447
 - \l@d@dld@ta 992,
1262, 1274, 1287, 1299, 1308, 1320
 - \l@d@leftbox
.. 945, 975, 985, 1806, 2044, 2047
 - \l@d@l@linenumR 225
 - \l@d@l@sln@te 994
 - \l@d@make@labels 1420
 - \l@d@make@labelsR 1408, 1425
 - \l@d@minpagelines
.... 1979, 2019, 2098, 2195, 2209
 - \l@d@numpstartsL
..... 837, 838, 840, 842, 857,
921, 1638, 1670, 1725, 1764,
1765, 1841, 1990, 1991, 2107, 2290
 - \l@d@numpstartsR
.. 47, 881, 882, 884, 886, 900,
942, 1638, 1671, 1728, 1764,
1765, 1842, 1990, 1991, 2108, 2339
 - \l@d@dold@bbl@set@language 1699
 - \l@d@dold@selectlanguage 1698, 1702, 1707
 - \l@d@pagefullfalse
.... 2186, 2222, 2227, 2234, 2239
 - \l@d@pagefulltrue 2186,
2221, 2226, 2233, 2238, 2317, 2360
 - \l@d@p@agingfalse 14, 736, 756
 - \l@d@p@agingtrue 750
 - \l@d@p@airingfalse 13, 740, 755
 - \l@d@p@airingtrue 735, 749
 - \l@d@p@scL 960, 965,
978, 1003, 1006, 1008, 1640,
1672, 1725, 1738, 1773, 1779,
1785, 1788, 1790, 1839, 2000,
2006, 2011, 2014, 2022, 2036,
2037, 2039, 2105, 2278, 2280,
2283, 2290, 2301, 2302, 2307,
2309, 2311, 2319, 2320, 2388, 2443
 - \l@d@p@scR 1032, 1037, 1050,
1641, 1673, 1728, 1741, 1774,
1780, 1796, 1799, 1801, 1840,
2001, 2007, 2015, 2023, 2067,
2068, 2070, 2106, 2327, 2329,
2332, 2339, 2348, 2349, 2352,
2354, 2356, 2362, 2363, 2396, 2445
 - \l@d@drd@ta 995,
1264, 1272, 1289, 1297, 1310, 1318
 - \l@d@rightbox
945, 1047, 1058, 1811, 2075, 2078
 - \l@d@drsn@te 996
 - \l@d@samepagefalse 2186,
2221, 2226, 2233, 2238, 2318, 2361
 - \l@d@samepagetrue
.... 2186, 2222, 2227, 2234, 2239
 - \l@d@setupmaxlinecounts .. 1651, 1668
 - \l@d@setuprawboxes 1643, 1667
 - \l@d@skipnumberfalse 1239
 - \l@d@skipnumbertrue 1135

- \l@dunhbox@line 995, 1011, 1014
- \l@dusedbabelfalse 1675, 1695
- \l@dusedbabeltrue 1675, 1697
- \l@duselanguage
 - 1687, 1783, 1794, 2032, 2063
- \l@dzeromaxlinecounts . . . 1651, 1669
- \l@dzeropenalties 909, 930, 1768, 1995
- \l@prev@nopbR 54, 2375
- \l@prev@pbR 53, 2374
- \l@pscL 1640
- \l@pscR 1640
- \label@refs
 - 1400, 1402, 1408, 1412, 1414, 1420
- \labelref@list 1411, 1414, 1443
- \labelref@listR 1394, 1399, 1402, 1439
- \languagename . . 1678, 1679, 1681–1684
- \last@page@num 354, 360
- \last@page@numR 340
- \lastbox 968, 1040
- \lastskip 617, 623
- \Lcolwidth
 - . . 4, 5, 16, 752, 851, 976, 985,
 - 1791, 1906, 1927, 1944, 1955, 1971
- \led@err@BadLeftRightPstarts . .
 - 29, 1765, 1991
- \led@err@LeftOnRightPage . . 32, 2128
- \led@err@LineationInNumbered . . 149
- \led@err@ManyLeftnotes 1486
- \led@err@ManyRightnotes 1493
- \led@err@ManySidenotes 1466
- \led@err@NumberingNotStarted . . . 91
- \led@err@NumberingShouldHaveStarted
 - 121
- \led@err@NumberingStarted 44
- \led@err@PendNoPstart 907, 928
- \led@err@PendNotNumbered . . . 904, 925
- \led@err@PstartInPstart . . . 827, 871
- \led@err@PstartNotNumbered . . 823, 867
- \led@err@RightOnLeftPage . . 32, 2135
- \led@err@TooManyPstarts
 - 24, 26, 839, 883
- \led@mess@NotesChanged 88
- \led@mess@SectionContinued
 - 115, 131, 139
- \led@nopbnumR 2379, 2380
- \led@nopbR 2378, 2380
- \led@pb@setting
 - 1869, 1877, 2220, 2224,
 - 2232, 2236, 2247, 2248, 2258, 2259
- \led@pbnumR 2377, 2380
- \led@pbR 2376, 2380
- \led@warn@BadAction 1137
- \led@warn@BadAdvancelineLine 385, 391
- \led@warn@BadAdvancelineSubline .
 - 371, 377
- \led@warn@BadLineation 166
- \led@warn@BadSetline 636
- \led@warn@BadSetlinenum 644
- \led@warn@DuplicateLabel 1427
- \ledgroupnotesL@false 2404
- \ledgroupnotesL@true 2392
- \ledgroupnotesR@false 2407
- \ledgroupnotesR@true 2400
- \ledllfill 995
- \lednopb 794
- \lednopbnum 2247, 2376
- \lednopbnumR 2258, 2376
- \lednopbR 794, 2378
- \ledpb 793
- \ledpbnum 2248
- \ledpbnumR 2259, 2376
- \ledpbR 793, 2376
- \ledRcol@false 1061
- \ledRcol@true 1030
- \ledRcolfalse 15, 767, 800
- \ledRcoltrue 784
- \ledrlfill 995, 1915, 1937
- \ledsavedprintlines 8, 1381
- \ledsectnomark 1002
- \ledsectnotoc 1001, 2035, 2066
- \ledstrutL 2044, 2047, 2113
- \ledstrutR 2075, 2078, 2113
- \ledthegoal 2194, 2208, 2264
- \leftlinenumR 225, 1262, 1274
- \leftpstartnumL 1282
- \leftpstartnumR 1282
- Leftside (environment) 5, 766
- \Leftsidehook 773, 778
- \Leftsidehookend 777, 778
- \line@list 724, 728
- \line@list@stuff 132
- \line@list@stuffR . . 62, 116, 140, 592
- \line@listR . 83, 243, 256, 568, 715, 719
- \line@margin 182, 1292
- \line@marginR 175, 1267, 1313
- \line@num . 357, 389, 390, 392, 410,
- 421, 422, 450, 575, 1094, 1441, 2295
- \line@numR 55,
- 232, 239, 294, 328, 347, 383,
- 384, 386, 403, 417, 418, 441,

561, 565, 579, 1076, 1148, 1157,
1246, 1248, 1250, 1251, 1437, 2344
`\lineation` 171, 172, 795
`\lineation*` 6, 171
`\lineationR` 6, 147, 173, 795
`\linenum@out`
 . 614, 620, 626, 632, 639, 647,
 652, 656, 1410, 1749, 1826, 2167
`\linenum@outR`
 . 589, 595, 597, 601, 602, 604,
 605, 608, 609, 616, 619, 625,
 631, 638, 646, 651, 655, 660,
 1398, 1754, 1830, 2170, 2376–2379
`\linenumberlist` 1247, 1251
`\linenumincrement` 6, 196
`\linenumincrement*` 6, 196
`\linenummargin` 175
`\linenumr@p` 1384, 1388, 1437, 1441
`\linenumrepR` 222, 232
`\linenumsep`
 . 227, 229, 1329, 1332, 1341, 1344
`\linesinpar@listL`
 248, 264, 576, 2141, 2144
`\linesinpar@listR`
 248, 260, 580, 2147, 2150
`\linesonpage@listL` 265, 584, 2154, 2157
`\linesonpage@listR` 261, 587, 2160, 2163
`\list@clear`
 . 256–261, 264, 265, 267, 832, 876
`\list@clearing@reg` 263
`\list@create`
 ... 243–246, 248–250, 1348, 1394
`\listcsxadd` 362, 2380–2383
`\lock@disp` 1207, 1211, 1216
`\lock@off` 499, 500, 508, 655, 656
`\lock@on` 651, 652

M

`\managestanza@modulo` 1610
`\maxchunks` 3, 1634
`\maxlinesinpar@list` 248, 267
`\memorydump` 7, 772, 789
`\memorydumpL` 126, 772
`\memorydumpR` 126, 789
`\message` 61
`\multiply` 1181, 1909, 1958

N

`\n@num` 536, 660
`\n@num@reg` 542

`\namebox` 960, 965, 1032,
 1037, 1618, 1738, 1741, 2278, 2327
`\NeedsTeXFormat` 2
`\new@line` 1011, 1014
`\new@lineL` 613, 995
`\new@lineR` 615
`\newbox` 805, 945, 946, 1619
`\newcommandx` 815, 859, 902, 923
`\newcounter` 99–102, 187,
 189, 191, 193, 808, 809, 811, 812
`\newif` 5, 39, 143, 144, 590, 764, 765,
 1336, 1541, 1675, 1722, 1734,
 1867, 1868, 2186, 2188, 2274, 2275
`\newlength` 1895, 1898
`\newmarks` 2384–2386
`\newnamebox` 1618, 1646, 1647
`\newnamecount` 1629, 1654
`\newsavebox` 1758, 1759
`\newwrite` 589, 2373
`\next@absline`
 1870, 1872, 1874, 2225–2227
`\next@abslineR`
 1871, 1873, 1875, 2237–2239
`\next@action` 284
`\next@actionline` 281, 283
`\next@actionlineR`
 . 273, 275, 1106, 1144, 1166, 1168
`\next@actionR` 276, 1107,
 1145, 1146, 1151, 1152, 1160, 1169
`\next@insert` 833
`\next@insertR`
 877, 1352, 1355, 1357, 1360, 1364
`\next@page@num` 361, 433
`\next@page@numR` 59, 297, 299, 351, 430
`\no@expands` 668, 691
`\noindent` 857, 900, 921, 942
`\normal@page@breakR` 52
`\normal@pars` 76, 836, 880
`\normalbfnoteX` 1514
`\notefontsetup` 2430
`\noteschanged@true`
 81, 84, 716, 725, 1354
`\notesXwidthliketwocolumns` 5
`\num@lines` 910, 1769, 1996
`\num@linesR` 804, 931, 1770, 1997
`\numberedpar@true` 852, 895
`\numberingRfalse` 75
`\numberingRtrue` 49, 110, 136
`\numberingtrue` 128
`\numberpstartfalse` 8

- `\numberpstarttrue` 8
- `\numdef` 1003, 1484, 1491,
1870, 1871, 2225, 2237, 2443, 2445
- `\numgdef` 1456, 1463, 2246, 2257
- `\numlabfont` 232
- `\numpagelinesL` 1979,
2042, 2055, 2059, 2195, 2248, 2463
- `\numpagelinesR`
1979, 2073, 2086, 2090, 2209, 2259
- O**
- `\old@otherlanguage` 1712
- `\old@startstanza` .. 774, 775, 797, 798
- `\oldchapter` 747, 757
- `\one@line` ... 965, 968, 995, 1011, 1014
- `\one@lineR` 804, 1037, 1040
- `\openout` 70, 597, 602, 605, 609
- `\otherlanguage` 1712, 1713
- P**
- `\p@pstartL` 854
- `\p@pstartR` 897
- `\PackageError` 21
- `\page@action` 298, 427, 555
- `\page@num` 279, 359, 1294
- `\page@numR` 252, 271, 349,
560, 565, 1146, 1269, 1315, 1479
- `\pagebreak` 1884
- `\pagegoal` 2272
- `\Pages` 5, 1983
- `pages` (environment) 5, 734
- `\pagetotal` 2120, 2194, 2208
- `pairs` (environment) 3, 734
- `\paperwidth` 1010, 1013
- `\par@line` 911, 1771, 1998
- `\par@lineR` 804, 932, 1772, 1999
- `\parbox` 1791, 1802
- `\parledgroup@` .. 967, 1039, 2384, 2410
- `\parledgroup@beforenotes@save` ..
..... 918, 939, 2470
- `\parledgroup@beforenotesL` 2468
- `\parledgroup@beforenotesR` 2468
- `\parledgroup@correction@notespacing`
..... 2046, 2077, 2457
- `\parledgroup@correction@notespacing@final`
..... 2313, 2358, 2438
- `\parledgroup@correction@notespacing@ini`
..... 2060, 2091, 2433, 2441
- `\parledgroup@notes@endL`
..... 2281, 2284, 2312, 2403
- `\parledgroup@notes@endR`
..... 2330, 2333, 2357, 2406
- `\parledgroup@notes@startL`
..... 967, 2387, 2403
- `\parledgroup@notes@startR`
..... 1039, 2387, 2403
- `\parledgroup@notespacing@correction`
..... 2428, 2460–2462
- `\parledgroup@notespacing@correction@accumulated`
..... 2434, 2440, 2461
- `\parledgroup@notespacing@correction@modulo`
..... 2435, 2462–2464
- `\parledgroup@notespacing@set@correction`
..... 1987, 2428
- `\parledgroup@series`
..... 2385, 2389, 2390,
2397, 2398, 2413, 2414, 2420, 2421
- `\parledgroup@type` 2386,
2389, 2390, 2397, 2398, 2411, 2418
- `\parledgroupnotespacing` . 2427, 2430
- `\parledgroupseries@` 2384
- `\parledgrouptrue` 10
- `\parledgrouptype@` 2384
- `\pausenumbering` 787
- `\pausenumberingR` 109, 787
- `\pend` 6, 771, 792, 828, 1597
- `\pendL` 771, 902
- `\pendR` 792, 872, 923
- `\prev@abslineverse`
..... 2246–2248, 2257–2259
- `\prev@nopbR` 2374
- `\prev@pbR` 2374
- `\prevgraf`
..... 910, 931, 1769, 1770, 1996, 1997
- `\print@columnseparator` .. 1810, 1847
- `\print@eledsectionL`
..... 990, 998, 1791, 2040
- `\print@eledsectionR` . 1065, 1802, 2071
- `\print@lineL` 980, 990
- `\print@lineR` 1052, 1065
- `\printlines` 1392
- `\printlinesR` 8, 1381
- `\ProcessOptions` 12
- `\protected@csxdef` 1523, 1534
- `\protected@edef` 853, 896
- `\protected@write` ... 1407, 1419, 1681
- `\ProvidesPackage` 3
- `\pst@rtedLfalse` 38
- `\pst@rtedLtrue` 129, 834
- `\pst@rtedRfalse` 40, 48, 78

- `\pst@rtedRtrue` 113, 137, 878
`\pstart` 6, 27, 31, 769, 791, 1599
`\pstartL` 769, 807
`\pstartnumfalse` 1329, 1334
`\pstartnumRfalse` 1341, 1346
`\pstartnumRtrue` 1337, 1778, 2351
`\pstartnumtrue` 1777, 2306
`\pstartR` 791, 807
- R**
- `\Rcolwidth`
 4, 5, 16, 753, 894, 1048, 1058,
 1802, 1907, 1928, 1945, 1956, 1972
`\read@linelist` 254, 593
`\rem@inder` 1251, 1253–1255
`\resetprevline@` 1827, 1831, 2296, 2345
`\resumenumbering` 788
`\resumenumberingR` 109, 788
`\rightlinenumR` 225, 1264, 1272
`\rightpstartnumL` 1282
`\rightpstartnumR` 1282
Rightside (environment) 5, 783
`\Rightsidehook` 778, 796
`\Rightsidehookend` 778, 801
`\rlap` 1264, 1272, 1289, 1297, 1310, 1318
`\Rlineflag` 8, 220, 232, 1384, 1388, 1429
`\rule` 1887
- S**
- `\savebox` 1791, 1802
`\sc@n@list` 1252, 1254
`\secdef` 762
`\section@num` 130–132
`\section@numR` 36,
 50, 61, 62, 69, 70, 114–116, 138–140
`\select@language` 1679, 1681–1684, 1718
`\selectlanguage` 1687
`\set@line` 669, 692, 713
`\set@line@action`
 291, 396, 405, 412, 435, 557
`\setl@dlp@rbox` 1472, 1487, 1489
`\setl@drp@rbox` 1474, 1482, 1494
`\setline` 634
`\setlinenum` 642
`\setnamebox` 842, 886, 1618
`\setnotepositionliketwocolumns@C`
 1901
`\setnotepositionliketwocolumns@L`
 1901
`\setnotepositionliketwocolumns@R`
 1901
`\setnotespositionliketwocolumns@C`
 1940
`\setnotespositionliketwocolumns@L`
 1918
`\setnotespositionliketwocolumns@R`
 1967
`\setpositionliketwocolumns@C` ...
 1901, 1935
`\setpositionliketwocolumns@L` ...
 1901, 1914
`\setpositionliketwocolumns@R` ...
 1901, 1963
`\setprintlines` 1382
`\setwidthliketwocolumns@C` 1901, 1922
`\setwidthliketwocolumns@L` 1901, 1901
`\setwidthliketwocolumns@R` 1901, 1952
`\shiftedpstartfalse` 7
`\shiftedpstarttrue` 6, 8, 9
`\shiftedversesfalse` 7
`\shiftedversestrue` 6
`\showlemma` 678, 701
`\sidenote@margin` 1449, 1452
`\sidenote@marginR` 1446, 1477
`\sidenotecontent@`
 . 1455, 1460, 1461, 1472, 1474,
 1482, 1483, 1487, 1489, 1490, 1494
`\sidenotemargin` 1446
`\sidenotemargin*` 1446
`\sidenotesep` 1461
`\skip` 2413, 2420
`\skip@lockoff` 500, 508
`\skipnumbering` 9, 659
`\skipnumbering@reg` 663
`\smash` 1887
`\splitbotmarks` 2410,
 2411, 2413, 2414, 2418, 2420, 2421
`\splitfirstmarks`
 967, 1039, 2389, 2390, 2397, 2398
`\splittopskip` 962, 1034
`\stanza@count` 1579, 1593, 1605
`\stanza@hang` 1581, 1613
`\stanza@modulo` 1579, 1608
`\stanzaindentbase`
 1559, 1569, 1606, 1609
`\startlock` 650
`\startstanzahook` 1577
`\startsub` 617
`\sub@action` 307, 456, 556

- `\sub@change` 60, 301, 302, 308
`\sub@lock` 1089
`\sub@lockR` 57, 316, 318, 320,
 323, 474, 480, 481, 483, 484,
 514, 515, 517, 1071, 1127, 1129,
 1130, 1132, 1188, 1228, 1230, 1232
`\sub@off` 625, 626
`\sub@on` 619, 620
`\subline@num` 234, 357, 375,
 376, 378, 408, 448, 1090, 1095, 1442
`\subline@numR` 235,
 239, 324, 328, 347, 369, 370,
 372, 401, 439, 562, 566, 1072,
 1077, 1148, 1155, 1242, 1243, 1438
`\sublinenumincrement` 6, 196
`\sublinenumincrement*` 6, 196
`\sublinenumr@p` . 1385, 1389, 1438, 1442
`\sublinenumrepR` 222, 235
`\sublines@false` 58, 305, 1117
`\sublines@true` 303, 1115
`\sublock@disp` 1190, 1194, 1199
`\symplinenum` 1384
`\sza@penalty` 1588, 1592
- T**
- `\temp` 1904, 1905, 1908–1911,
 1924, 1925, 1930–1932, 1941,
 1943, 1946–1949, 1953, 1954,
 1957–1960, 1968, 1970, 1973–1976
`\temp@` 1003, 1004
`\temp@spacing` 2430, 2431
`\tempa` 1942, 1944–1946, 1969, 1971–1973
`\textwidth` 17, 19, 752, 753
`\theledlanguageL` . . . 1687, 1783, 2032
`\theledlanguageR` . . . 1687, 1794, 2063
`\thepage` 614, 616, 1408, 1420
`\thepstart` 770, 790
`\thepstartL`
 . 8, 770, 810, 847, 854, 1328, 1333
`\thepstartR`
 . 8, 790, 813, 890, 897, 1340, 1345
`\thr@@` . . 483, 492, 515, 522, 1122, 1130
`\topskip` 2120
- U**
- `\unhbox` 1623,
 1806, 1811, 2044, 2047, 2075, 2078
- `\unhnamebox` 1618
`\unvbox` 968, 1040, 1625
`\unvnamebox` 1618
`\usebox` 1808, 1813
`\usernamecount` 1580, 1587, 1629, 1661,
 2011, 2280, 2283, 2307, 2309,
 2329, 2332, 2352, 2354, 2388, 2396
- V**
- `\value` 831, 875, 1604,
 1762, 1763, 1985, 1986, 2293, 2342
`\vbadness` 961, 1033
`\vbfnoteX` 1524, 1535
`\vbox` 842, 886, 1791, 1802
`\vl@dbfnote` 1504, 1508
`\vsplit` 965, 1037
- W**
- `\wd` 995
`\widthliketwocolumns` 4
`\widthliketwocolumnstrue` 11
`\WithSuffix` 171, 216–219, 1446
`\writtenlinesLfalse` 2002, 2291
`\writtenlinesLtrue` 2288
`\writtenlinesRfalse` 2003, 2340
`\writtenlinesRtrue` 2337
- X**
- `\x@lemma` 680–682, 703–705
`\xifinlist` 978,
 1004, 1050, 1785, 1796, 2039, 2070
`\xifinlistcs` 1872–
 1875, 1878–1881, 2221, 2222,
 2226, 2227, 2233, 2234, 2238, 2239
`\Xnoteswidthliketwocolumns` 5
`\xpg@main@language` 1719, 1720
`\xright@appenditem`
 429, 430, 432, 433,
 437, 444, 446, 453, 458, 460,
 462, 465, 467, 469, 477, 479,
 488, 511, 513, 520, 539, 540,
 550, 564, 576, 580, 584, 587,
 1437, 1441, 1504, 1508, 1524, 1535
- Z**
- `\zz@@@` 1400, 1412

Change History

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

	<code>\Rightsidehookend:</code>	Added	Possibility to number the	
	<code>\Leftsidehook, \Leftsidehookend,</code>		<code>pstart</code> with the commands	
	<code>\Rightsidehook and \Rightsidehookend</code>		<code>\numberpstarttrue</code>	1
	35	<code>\ifledRcol:</code> Moved <code>\iflledRcol</code>	
	<code>\sublinenumrepR:</code>	Added	and <code>\ifnumberingR</code> to <code>eledmac</code>	13
	<code>\linenumrepR and \sublinenumrepR</code>	v0.9.1		
	20	General: The numbering of the	
v0.3.a	General: Minor <code>\linenummargin</code>		<code>pstarts</code> restarts on each	
	fix	1	<code>\beginnumbering</code>	1
v0.3.b	General: Improved parallel page			
	balancing	1	General: Debug : with <code>\Columns</code> ,	
v0.3.c	General: Compatibility with Poly-		the hanging indentation now	
	glossia	1	runs on the left columns and the	
v0.3.a	General: Compatibility with Poly-		hanging symbol is shown only	
	glossia	1	when <code>\stanza</code> is used.	1
v0.3a	<code>\line@marginR:</code> Don't just		v0.9.3	
	set <code>\line@marginR</code> in		General: <code>\thepstartL</code> and	
	<code>\linenummargin</code>	18	<code>\thepstartR</code> use now	
v0.3b	<code>\Pages:</code> Added <code>\l@dminpagelines</code>		<code>\bfseries</code> and not <code>\bf</code> , which	
	calculation for succeeding page		is deprecated and makes con-	
	pairs	72	flicts with memoir class.	1
v0.4	General: No more <code>ledparpatch</code> . All		v1.0	
	patches are now in the main		General: Compatibility with <code>eled-</code>	
	file.	1	<code>mac</code> . Change name to <code>eledpar</code> . .	1
v0.5	General: Corrections about		Debug in lineation by <code>pstart</code> . .	17
	<code>\section</code> and other titles in		v1.0.1	
	numbered sections	1	General: Correction on <code>\numberonlyfirstinline</code>	
v0.6	General: Be able to use <code>\chapter</code> in		with lineation by <code>pstart</code> or by	
	parallel pages.	1	page.	1
v0.7	General: Option 'shiftedverses'		v1.1	
	which make there is no blank		General: <code>Shiftedverses</code> becomes	
	inequal length.	1	<code>shiftedpstarts</code>	1
v0.8	General: Possibility to have a sym-		<code>\pstartR:</code> Add <code>\labelpstarttrue</code>	
	bol on each hanging of verses,		(from <code>eledmac</code>).	37
	like in the french typogra-		v1.1.1	
	phy. Redefine the commande		<code>\pstartR:</code> Correct <code>\pstartR</code> bug in-	
	<code>\hangingsymbol</code> to define the		troduced by 1.1.	37
	character.	1	v1.1.2	
v0.9	General: Possibility to number		<code>\affixside@noteR:</code> Remove spuri-	
	<code>\pstart</code>	8	ous space between line number	
			and line content	54
			v1.2	
			General: Support for <code>\led{section}</code>	
			commands in parallel texts. . . .	1
			v1.2.1	
			<code>\initnumbering@sectcountR:</code> For	
			the right section, the counter is	
			defined only once.	16
			v1.3	
			General: Manage RTL language. .	33

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

<code>\pendL</code> : As in <code>eledmac</code> , <code>\pendL</code> can have an optional argument. . .	39	<code>\flag@end</code> : <code>\flag@start</code> and <code>\flag@end</code> are now defined only one time for <code>eledmac</code> and <code>eledpar</code>	31
<code>\pendR</code> : As in <code>eledmac</code> , <code>\pendR</code> can have an optional argument. . .	40	<code>\lineation*</code> : Add <code>\lineation*</code> .	18
<code>\print@columnseparator</code> : Move some code of <code>\Columns</code> to <code>\print@columnseparator</code> . . .	65	v1.8.3	
<code>\pstartR</code> : As in <code>eledmac</code> , <code>\pendL</code> and <code>\pendR</code> can have an optional argument.	37	General: Add <code>\noeledxxx</code> , as in <code>eledmac</code>	1
<code>\sidenotemargin*</code> : <code>\sidenotemargin</code> is now directly defined in <code>eledmac</code> to be able to manage <code>eledpar</code>	54	<code>\doinsidelineRhook</code> : Added <code>\dolineLhook</code> , <code>\dolineRhook</code> , <code>\doinsidelineLhook</code> and <code>\doinsidelineRhook</code>	42
Add <code>\sidenotemargin*</code>	54	<code>\Pages</code> : Debug blank pages when using optional argument in the last <code>\pend</code>	69
<code>\theledlanguageR</code> : Correct left/right language setting with <code>polyglossia</code>	62	<code>\resumenumberingR</code> : Debug <code>\resumenumberingR</code>	16
v1.8.1		v1.9.0	
<code>\do@lineL</code> : Fix a bug with critical notes at the beginning of a page, (maybe added by v1.8.0) (?). .	41	General: Add <code>\AtBeginPairs</code> macro.	4
<code>\do@lineR</code> : Fix a bug with critical notes at the beginning of a page, added by v1.8.0 (?).	42	Compatibility with <code>\Xnoteswidthliketwocolumns</code> and <code>\notesXwidthliketwocolumns</code>	1
v1.8.2		<code>\ifwidthliketwocolumns</code> : Added <code>widthliketwocolumns</code> option . .	13
General: Debug <code>\eledxxx</code> with some paper sizes	1	<code>\theledlanguageR</code> : Debug left/right language switching with <code>polyglossia</code> . Don't write in <code>.aux</code> file when setting left/right lines.	62
Debug left and side note (bugs added by 1.8.0)	1	v1.9.1	
<code>\eledpar@error</code> : Errors specific to <code>eledpar</code> send to <code>eledpar</code> handbook	14	<code>\ifledRcol</code> : Moved <code>\ifl@dpaging</code> to <code>eledmac</code>	13