

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

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

Abstract

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

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

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

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

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

Contents

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

*This file (**eledpar.dtx**) has version number v1.8.2, last revised 2014/08/15.

[†]herries dot press at earthlink dot net

[‡]maieul at maieul dot net

5 Left and right texts	7
6 Numbering text lines and paragraphs	8
7 Verse	10
8 Side notes	12
9 Parallel ledgroups	12
9.1 Parallel ledgroups and <code>setspace</code> package	13
10 Sectioning commands	14
11 Implementation overview	14
12 Preliminaries	14
12.1 Messages	15
13 Sectioning commands	16
14 Line counting	18
14.1 Choosing the system of lineation	18
14.2 Line-number counters and lists	22
14.3 Reading the line-list file	22
14.4 Commands within the line-list file	23
14.5 Writing to the line-list file	31
15 Marking text for notes	34
16 Parallel environments	36
17 Paragraph decomposition and reassembly	37
17.1 Boxes, counters, <code>\pstart</code> and <code>\pend</code>	38
17.2 Processing one line	41
17.3 Line and page number computation	44
17.4 Line number printing	47
17.5 Pstart number printing in side	49
17.6 Add insertions to the vertical list	51
17.7 Penalties	51
17.8 Printing leftover notes	52
18 Footnotes	53
18.1 Normal footnote formatting	53
19 Cross referencing	53
20 Side notes	55

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

List of Figures

1 Introduction

The EDMAC macros [LW90] for typesetting critical editions of texts have been available for use with TeX for some years. Since EDMAC became available there had been a small but constant demand for a version of EDMAC that could be used with LaTeX. 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}
...
\Columns
\end{pairs}
```

There is no required pagebreak before or after the columns.

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

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

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

<code>\columnrulewidth</code>	The macro <code>\columnseparator</code> is called between each left/right pair of lines.
<code>\columnseparator</code>	By default it inserts a vertical rule of width <code>\columnrulewidth</code> . As this is initially defined to be 0pt the rule is invisible. For a visible rule between the columns you could try: <code>\setlength{\columnrulewidth}{0.4pt}</code> You can also modify <code>\columnseparator</code> if you want more control.
<code>\columnspan</code>	By default, columns are positioned to the right of the page. However, you use <code>\columnspan{L}</code> to align them to the left, or <code>\columnspan{C}</code> to center them.
<code>\columnspan</code>	When you use <code>\stanza</code> , the visible rule may shift when a verse has a hanging indent. To prevent shifting, use <code>\setstanzaindents</code> outside the <code>Leftside</code> or <code>Rightside</code> environment.
<code>\beforecolumnseparator</code> <code>\aftercolumnseparator</code>	By default, the spaces around column separator are the same as the space: <ul style="list-style-type: none"> • On the left of columns, if columns are aligned right. • On the right of columns, if columns are aligned left. • On both the Left and Right columns, if columns are centered.

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

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

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

4 Facing pages

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

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

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

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

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

\newcommand*{\goalfraction}{0.9}

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

\renewcommand*{\goalfraction}{0.8}

5 Left and right texts

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

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

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

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

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

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

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

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

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

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

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

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

6 Numbering text lines and paragraphs

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

```
\beginnumbering
<text>
```


`\endnumbering`

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

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

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

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

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

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

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

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

following code in your preamble:

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

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

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

7 Verse

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

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

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

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

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

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

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

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

```

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

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

  \end{astanza}
  ...

```

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

```

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

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

  \end{astanza}
  ...

```

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

```

\renewcommand{\hangingsymbol}{[,]}

```

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

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

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

8 Side notes

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

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

9 Parallel ledgroups

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

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

See the following example:

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

```

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

```

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

```

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

```

9.1 Parallel ledgroups and **setspace** package

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

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

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

10 Sectioning commands

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

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

11 Implementation overview

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

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

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

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

12 Preliminaries

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

```
1 *code
2 \NeedsTeXFormat{LaTeX2e}
3 \ProvidesPackage{eledpar}[2014/08/15 v1.8.2 eledmac extension for parallel texts]%
4
```

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

`\ifshiftedpstarts`

```

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

```

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

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

```

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

```

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

```

\l@dpairing
\l@dpaging
\ledRcol
16 \newdimen\Lcolwidth
17 \Lcolwidth=0.45\textwidth
18 \newdimen\Rcolwidth
19 \Rcolwidth=0.45\textwidth
20

```

12.1 Messages

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

`\eledpar@error`

```

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

```

`\led@err@BadLeftRightPstarts`

```

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

```

```

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

```

13 Sectioning commands

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

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

```

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

```

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

```

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

```

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

```

42 \newcommand*{\beginnumberingR}{%
43 \ifnumberingR
44 \led@err@NumberingStarted
45 \endnumberingR
46 \fi
47 \global\l@dnumpstartsR \z@
48 \global\pst@rtedRfalse
49 \global\numberingRtrue
50 \global\advance\section@numR \@ne
51 \global\absline@numR \z@
52 \gdef\normal@page@breakR{}
53 \gdef\l@prev@pbR{}
54 \gdef\l@prev@nopbR{}
55 \global\line@numR \z@
56 \global\@lockR \z@
57 \global\sub@lockR \z@
58 \global\sublines@false
59 \global\let\next@page@numR\relax
60 \global\let\sub@change\relax
61 \message{Section \the\section@numR R}%
62 \line@list@stuffR{\jobname.\extensionchars\the\section@numR R}%

```



```

63 \l@dend@stuff
64 \setcounter{pstartR}{1}
65 \begingroup
66 \initnumbering@sectcountR
67 \gdef\eled@sectionsR@@{}
68 \makeatletter\inputIfFileExists{\jobname.eledsec\the\section@numR R}{\makeatother
69 \immediate\openout\eled@sectioningR@out=\jobname.eledsec\the\section@numR R\relax
70 }

```

`\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 `\ifpstartL` 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.

```

71 \def\endnumberingR{%
72   \ifnumberingR
73     \global\numberingRfalse
74     \normal@pars
75     \ifl@pairing
76       \global\pstartRfalse
77     \else
78       \ifx\insertlines@listR\empty\else
79         \global\noteschanged@true
80       \fi
81       \ifx\line@listR\empty\else
82         \global\noteschanged@true
83       \fi
84     \fi
85     \ifnoteschanged@
86       \led@mess@NotesChanged
87     \fi
88   \else
89     \led@err@NumberingNotStarted
90   \fi
91 \endgroup
92 \immediate\closeout\eled@sectioningR@out
93 }
94

```

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

```

95 \newcounter{chapterR}
96 \newcounter{sectionR}
97 \newcounter{subsectionR}
98 \newcounter{subsubsectionR}
99 \newcommand{\initnumbering@sectcountR}{
100   \let\c@chapter\c@chapterR
101   \let\c@section\c@sectionR

```

```

102    \let\c@subsection\c@subsectionR
103    \let\c@subsubsection\c@subsubsectionR
104 }

\pausenumberingR These are the right text equivalents of \pausenumbering and \resumenumbering.
\resumenumberingR
105 \newcommand*{\pausenumberingR}{%
106   \endnumberingR\global\numberingRtrue}
107 \newcommand*{\resumenumberingR}{%
108   \ifnumberingR
109     \global\pst@rtedRtrue
110     \global\advance\section@numR \@ne
111     \led@mess@SectionContinued{\the\section@numR R}%
112     \line@list@stuffR{\jobname.\extensionchars\the\section@numR R}%
113     \l@dend@stuff
114     \initnumbering@sectcmd
115   \else
116     \led@err@numberingShouldHaveStarted
117     \endnumberingR
118     \beginnumberingR
119   \fi}
120

\memorydumpL \memorydump is a shorthand for \pausenumbering\resumenumbering. This will
\memorydumpR clear the memorised stuff for the previous chunks while keeping the numbering
going.
121 \newcommand*{\memorydumpL}{%
122   \endnumbering
123   \numberingtrue
124   \global\pst@rtedLtrue
125   \global\advance\section@num \@ne
126   \led@mess@SectionContinued{\the\section@num}%
127   \line@list@stuff{\jobname.\extensionchars\the\section@num}%
128   \l@dend@stuff}
129 \newcommand*{\memorydumpR}{%
130   \endnumberingR
131   \numberingRtrue
132   \global\pst@rtedRtrue
133   \global\advance\section@numR \@ne
134   \led@mess@SectionContinued{\the\section@numR R}%
135   \line@list@stuffR{\jobname.\extensionchars\the\section@numR R}%
136   \l@dend@stuff}
137

```

14 Line counting

14.1 Choosing the system of lineation

Sometimes you want line numbers that start at 1 at the top of each page; sometimes you want line numbers that start at 1 at each `\pstart`; other times you want line

numbers that start at 1 at the start of each section and increase regardless of page breaks. `eledpar` lets you choose different schemes for the left and right texts.

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

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

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

`\ifbypage@R`

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

`\bypage@Rfalse`

```

138 \newif\ifbypage@R
139 \newif\ifbypstart@R
140 \bypage@Rfalse
141 \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`.

```

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

```

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

```

166 \WithSuffix\newcommand\lineation*[1]{%
167   \lineation{#1}%
168   \lineationR{#1}%
169 }%

```

`\linenummargin` You call `\linenummargin{<word>}` to specify which margin you want your right

`\line@marginR` 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.

```

170 \newcount\line@marginR
171 \renewcommand*{\linenummargin}[1]{%
172   \l@getline@margin{#1}%
173   \ifnum\l@dttempcntb>\m@ne
174     \ifledRcol
175       \global\line@marginR=\l@dttempcntb
176     \else
177       \global\line@margin=\l@dttempcntb
178     \fi
179   \fi}}

```

By default put right text numbers at the right.

```

180 \line@marginR=\@ne
181

```

`\c@firstlinenumR` The following counters tell `eledmac` which right text lines should be printed with
`\c@linenumincrementR` 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.

```

182 \newcounter{firstlinenumR}
183 \setcounter{firstlinenumR}{5}
184 \newcounter{linenumincrementR}
185 \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.

```

186 \newcounter{firstsublinenumR}
187 \setcounter{firstsublinenumR}{5}
188 \newcounter{sublinenumincrementR}
189 \setcounter{sublinenumincrementR}{5}
190

```

`\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 `\provideing` them. The starred
`\firstsublinenum` versions are specific to `eledpar`.
`\sublinenumincrement` 191 \providecommand*{\firstlinenum}{}
`\firstlinenum*` 192 \providecommand*{\linenumincrement}{}
`\linenumincrement*` 193 \providecommand*{\firstsublinenum}{}
`\firstsublinenum*` 194 \providecommand*{\sublinenumincrement}{}
`\sublinenumincrement*` 195 \renewcommand*{\firstlinenum}[1]{%
 196 \ifledRcol \setcounter{firstlinenumR}{#1}%

```

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

```

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

```

215 \newcommand*{\Rlineflag}{R}
216

```

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

```

217 \newcommand*{\linenumrepR}[1]{\@arabic{#1}}
218 \newcommand*{\sublinenumrepR}[1]{\@arabic{#1}}
219

```

`\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`.

```

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

```

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.

```
234 \newcount\line@numR
235 \newcount\subline@numR
236 \newcount\absline@numR
237
```

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

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

```
238 \list@create{\line@listR}
239 \list@create{\insertlines@listR}
240 \list@create{\actionlines@listR}
241 \list@create{\actions@listR}
242
```

`\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.

```
243 \list@create{\linesinpar@listL}
244 \list@create{\linesinpar@listR}
245 \list@create{\maxlinesinpar@list}
246
```

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

```
247 \newcount\page@numR
248
```

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.

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

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

```
250   \ifledRcol
251     \list@clear{\line@listR}%
252     \list@clear{\insertlines@listR}%
```

```

253 \list@clear{\actionlines@listR}%
254 \list@clear{\actions@listR}%
255 \list@clear{\linesinpar@listR}%
256 \list@clear{\linesonpage@listR}
257 \else
258 \list@clearing@reg
259 \list@clear{\linesinpar@listL}%
260 \list@clear{\linesonpage@listL}%
261 \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).

```
262 \list@clear{\maxlinesinpar@list}
```

Now get the file and interpret it.

```

263 \get@linelistfile{#1}%
264 \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.

```

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

```

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 `\@nl` what it does depends on whether right text is being processed.

```

283 \newcommand{\@nl@regR}{%
284   \ifx\l@dchset@num\relax \else
285     \advance\absline@numR \@ne
286     \set@line@action
287     \let\l@dchset@num\relax
288     \advance\absline@numR \m@ne
289     \advance\line@numR \m@ne%    % do we need this?
290   \fi
291   \advance\absline@numR \@ne
292   \ifx\next@page@numR\relax \else
293     \page@action
294     \let\next@page@numR\relax
295   \fi
296   \ifx\sub@change\relax \else
297     \ifnum\sub@change>\z@
298       \sublines@true
299     \else
300       \sublines@false
301     \fi
302     \sub@action
303     \let\sub@change\relax
304   \fi
305   \ifcase\@lockR
306   \or
307     \@lockR \tw@
308   \or\or
309     \@lockR \z@
310   \fi
311   \ifcase\sub@lockR
312   \or
313     \sub@lockR \tw@
314   \or\or
315     \sub@lockR \z@
316   \fi
317   \ifsublines@
318     \ifnum\sub@lockR<\tw@
319       \advance\subline@numR \@ne
320     \fi
321   \else
322     \ifnum\@lockR<\tw@
323       \advance\line@numR \@ne \subline@numR \z@
324     \fi
325   \fi}
326
327 \renewcommand*{\@nl}[2]{%
```



```

328 \fix@page{#1}%
329 \ifledRcol
330   \@nl@regR
331 \else
332   \@nl@reg
333 \fi}
334

```

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

```

\fix@page 335 \newcount\last@page@numR
336   \last@page@numR=-10000
337 \renewcommand*{\fix@page}[1]{%
338   \ifledRcol
339     \ifnum #1=\last@page@numR
340     \else
341       \ifbypage@R
342         \line@numR \z@ \subline@numR \z@
343         \fi
344         \page@numR=#1\relax
345         \last@page@numR=#1\relax
346         \def\next@page@numR{#1}%
347       \fi
348     \else
349       \ifnum #1=\last@page@num
350       \else
351         \ifbypage@
352         \line@num \z@ \subline@num \z@
353         \fi
354         \page@num=#1\relax
355         \last@page@num=#1\relax
356         \def\next@page@num{#1}%
357         \listcsxadd{normal@page@break}{\the\absline@num}
358       \fi
359     \fi}
360

```

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

```

361 \renewcommand*{\@adv}[1]{%
362   \ifsublines@
363     \ifledRcol
364       \advance\subline@numR by #1\relax
365       \ifnum\subline@numR<\z@
366         \led@warn@BadAdvancelineSubline
367         \subline@numR \z@
368       \fi
369     \else
370       \advance\subline@num by #1\relax
371       \ifnum\subline@num<\z@

```

```

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

```

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

```

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

```

\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.

```

410 \renewcommand*{\l@d@set}[1]{%
411   \ifledRcol
412     \line@numR=#1\relax

```

```

413 \advance\line@numR \@ne
414 \def\l@dchset@num{#1}
415 \else
416 \line@num=#1\relax
417 \advance\line@num \@ne
418 \def\l@dchset@num{#1}
419 \fi}
420 \let\l@dchset@num\relax
421

```

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

```

422 \renewcommand*{\page@action}{%
423 \ifledRcol
424 \xright@appenditem{\the\absline@numR}\to\actionlines@listR
425 \xright@appenditem{\next@page@numR}\to\actions@listR
426 \else
427 \xright@appenditem{\the\absline@num}\to\actionlines@list
428 \xright@appenditem{\next@page@num}\to\actions@list
429 \fi}

```

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

```

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

```

`\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.

```

451 \renewcommand*{\sub@action}{%
452 \ifledRcol
453 \xright@appenditem{\the\absline@numR}\to\actionlines@listR

```

```

454 \ifsublines@
455 \xright@appenditem{-1001}\to\actions@listR
456 \else
457 \xright@appenditem{-1002}\to\actions@listR
458 \fi
459 \else
460 \xright@appenditem{\the\absline@num}\to\actionlines@list
461 \ifsublines@
462 \xright@appenditem{-1001}\to\actions@list
463 \else
464 \xright@appenditem{-1002}\to\actions@list
465 \fi
466 \fi}
467

```

`\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.

```

468 \newcount\@lockR
469 \newcount\sub@lockR
470
471 \newcommand*{\do@lockonR}{%
472 \xright@appenditem{\the\absline@numR}\to\actionlines@listR
473 \ifsublines@
474 \xright@appenditem{-1005}\to\actions@listR
475 \ifnum\sub@lockR=\z@
476 \sub@lockR \@ne
477 \else
478 \ifnum\sub@lockR=\thr@@
479 \sub@lockR \@ne
480 \fi
481 \fi
482 \else
483 \xright@appenditem{-1003}\to\actions@listR
484 \ifnum\@lockR=\z@
485 \@lockR \@ne
486 \else
487 \ifnum\@lockR=\thr@@
488 \@lockR \@ne
489 \fi
490 \fi
491 \fi}
492
493 \renewcommand*{\do@lockon}{%
494 \ifx\next\lock@off
495 \global\let\lock@off=\skip@lockoff
496 \else
497 \ifledRcol
498 \do@lockonR
499 \else

```

```

500      \do@lockonL
501      \fi
502  \fi}

```

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

```

\do@lockoff 503
\do@lockoffR 504
\skip@lockoff 505 \newcommand{\do@lockoffR}{%
506   \xright@appenditem{\the\absline@numR}\to\actionlines@listR
507   \ifsublines@
508     \xright@appenditem{-1006}\to\actions@listR
509     \ifnum\sub@lockR=\tw@
510       \sub@lockR \thr@@
511     \else
512       \sub@lockR \z@
513     \fi
514   \else
515     \xright@appenditem{-1004}\to\actions@listR
516     \ifnum\@lockR=\tw@
517       \@lockR \thr@@
518     \else
519       \@lockR \z@
520     \fi
521   \fi}
522
523 \renewcommand*{\do@lockoff}{%
524   \ifledRcol
525     \do@lockoffR
526   \else
527     \do@lockoffL
528   \fi}
529 \global\let\lock@off=\do@lockoff
530

```

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

```

531 \providecommand*\n@num{}
532 \renewcommand*\n@num{%
533   \ifledRcol
534     \xright@appenditem{\the\absline@numR}\to\actionlines@listR
535     \xright@appenditem{-1007}\to\actions@listR
536   \else
537     \n@num@reg
538   \fi}
539

```

`\@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`.

```
540 \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.

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

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

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

```
559 \xright@appenditem%
560   {\the\page@numR|\the\line@numR}%
561   \ifsublines@ \the\subline@numR \else 0\fi}%
562   \the\endpage@num|\the\endline@num}%
563   \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.

```
564 #2
565 \else
```

And when not in right text

```
566 \ref@reg{#1}{#2}%
567 \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.

```
568 \providecommand*\@pend}[1]{}
569 \renewcommand*\@pend}[1]{%
570 \ifbypstart@global\line@num=0\fi%
571 \xright@appenditem{#1}\to\linesinpar@listL}
572 \providecommand*\@pendR}[1]{}
573 \renewcommand*\@pendR}[1]{%
574 \ifbypstart@Rglobal\line@numR=0\fi
575 \xright@appenditem{#1}\to\linesinpar@listR}
576
```

`\@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.

```
577 \providecommand*\@lopL}[1]{}
578 \renewcommand*\@lopL}[1]{%
579 \xright@appenditem{#1}\to\linesonpage@listL}
580 \providecommand*\@lopR}[1]{}
581 \renewcommand*\@lopR}[1]{%
582 \xright@appenditem{#1}\to\linesonpage@listR}
583
```

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`.

```
584 \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 585 \newif\iffirst@linenum@out@R
586 \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.

```
587 \newcommand*\line@list@stuffR}[1]{%
588 \read@linelist{#1}%
```

```

589 \iffirst@linenum@out@R
590 \immediate\closeout\linenum@outR
591 \global\first@linenum@out@Rfalse
592 \immediate\openout\linenum@outR=#1
593 \else
594 \if@minipage%
595 \if@ledgroup%
596 \closeout\linenum@outR
597 \openout\linenum@outR=#1
598 \else
599 \immediate\closeout\linenum@outR
600 \immediate\openout\linenum@outR=#1\relax
601 \fi
602 \else
603 \closeout\linenum@outR%
604 \openout\linenum@outR=#1\relax%
605 \fi
606 \fi}
607

```

`\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.

```

608 \newcommand*{\new@lineL}{%
609 \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.

```

610 \newcommand*{\new@lineR}{%
611 \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.

```

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

```


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

```
625 \renewcommand*{\advanceline}[1]{%
626   \ifledRcol \write\linenum@outR{\string\@adv[#1]}%
627   \else      \write\linenum@out{\string\@adv[#1]}%
628   \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.

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

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

`\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.

```
645 \renewcommand*{\startlock}{%
646   \ifledRcol \write\linenum@outR{\string\lock@on}%
647   \else      \write\linenum@out{\string\lock@on}%
648   \fi}
649 \def\endlock{%
650   \ifledRcol \write\linenum@outR{\string\lock@off}%
651   \else      \write\linenum@out{\string\lock@off}%
652   \fi}
653
```

`\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.

```
654 \renewcommand*{\skipnumbering}{%
655   \ifledRcol \write\linenum@outR{\string\n@num}%
656   \advanceline{-1}%

```

```

657 \else
658   \skipnumbering@reg
659 \fi}
660

```

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.

```

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

```

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

```

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

`\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`.

```

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

16 Parallel environments

The initial set up for parallel processing is deceptively simple.

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

```
chapterinpages 729 \newenvironment{pairs}{%
730   \l@dpairingtrue
731   \l@dpagingfalse
732   \initnumbering@sectcmd
733 }{%
734   \l@dpairingfalse
735 }
```

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.

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

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

```
754 \newif\ifinstanzaL
755 \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.

```
756 \newenvironment{Leftside}{%
757   \ledRcolfalse
758   \setcounter{pstartL}{1}
```

```

759 \let\pstart\pstartL
760 \let\thepstart\thepstartL
761 \let\pend\pendL
762 \let\memorydump\memorydumpL
763 \Leftsidehook
764 \let\old@startstanza\@startstanza
765 \def\@startstanza[##1]{\global\instanzaLtrue\old@startstanza[##1]}
766 }{
767     \Leftsidehookend}

```

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

```

\Rightsidehook 768 \newcommand*\Leftsidehook{}
\Rightsidehookend 769 \newcommand*\Leftsidehookend{}
770 \newcommand*\Rightsidehook{}
771 \newcommand*\Rightsidehookend{}
772

```

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.

```

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

```

17 Paragraph decomposition and reassembly

In order to be able to count the lines of text and affix line numbers, we add an extra stage of processing for each paragraph. We send the paragraph into a box register, rather than straight onto the vertical list, and when the paragraph ends

we slice the paragraph into its component lines; to each line we add any notes or line numbers, add a command to write to the line-list, and then at last send the line to the vertical list. This section contains all the code for this processing.

17.1 Boxes, counters, `\pstart` and `\pend`

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

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

```
794 \newcount\num@linesR
795 \newbox\one@lineR
796 \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 begining of the `\Pages` command.

```
797
798 \newcounter{pstartL}
799 \newcounter{pstartLold}
800 \renewcommand{\thepstartL}{\bfseries\@arabic\c@pstartL}. }
801 \newcounter{pstartR}
802 \newcounter{pstartRold}
803 \renewcommand{\thepstartR}{\bfseries\@arabic\c@pstartR}. }
804
805 \newcommandx*{\pstartL}[1][1]{%
806   \if@nobreak%
807     \let\@oldnobreak\@nobreaktrue%
808   \else%
809     \let\@oldnobreak\@nobreakfalse%
810   \fi%
811   \@nobreaktrue%
812   \ifnumbering \else%
813     \led@err@PstartNotNumbered%
```

```

814 \beginnumbering%
815 \fi%
816 \ifnumberedpar%
817 \led@err@PstartInPstart%
818 \pend%
819 \fi%

```

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

```

820 \ifpst@rtedL\else%
821 \setcounter{pstartLold}{\value{pstartL}}%
822 \list@clear{\inserts@list}%
823 \global\let\next@insert=\empty%
824 \global\pst@rtedLtrue%
825 \fi%
826 \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.

```

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

```

Dump the optional arguments

```

845 \ifstrempy{#1}%
846 {}%
847 {\csgdef{before@pstartL@the\l@dnumpstartsL}{\noindent#1}}%
848 }

849 \newcommandx*{\pstartR}[1][1]{%
850 \if@nbreak%
851 \let\@oldnbreak\@nbreaktrue%
852 \else%
853 \let\@oldnbreak\@nbreakfalse%
854 \fi%
855 \@nbreaktrue%

```

```

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

```

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

```

892 \newcommand*{\pendL}[1][1]{%
893   \ifnumbering \else%
894     \led@err@PendNotNumbered%
895   \fi%
896   \ifnumberedpar@ \else%
897     \led@err@PendNoPstart%
898   \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.

```
899 \l@dzzeropenalties%
900 \endgraf\global\num@lines=\prevgraf\egroup%
901 \global\par@line=0%
```

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

```
902 \endgroup%
903 \ignorespaces%
904 \@oldnbreak%
905 \ifnumberpstart%
906 \addtocounter{pstartL}{1}%
907 \fi
908 \parledgroup@beforenotes@save{L}%
```

Dump content of the optional argument.

```
909 \ifstrepty{#1}%
910 {}%
911 {\csgdef{after@pendL@the\l@dnumpstartsL}{\noindent#1}}%
912 }
```

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

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

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@leftbox  A line of left text will be put in the box \l@leftbox, and analagously for a line
\l@drighbox of right text.

935 \newbox\l@leftbox
936 \newbox\l@drighbox
937

\countLline We need to know the number of lines processed.
\countRline
938 \newcount\countLline
939 \countLline \z@
940 \newcount\countRline
941 \countRline \z@
942

\@donereallinesL We need to know the number of ‘real’ lines output (i.e., those that have been input
\@donetotallinesL by the user), and the total lines output (which includes any blank lines output for
\@donereallinesR synchronisation).
\@donetotallinesR
943 \newcount\@donereallinesL
944 \newcount\@donetotallinesL
945 \newcount\@donereallinesR
946 \newcount\@donetotallinesR
947

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

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

```

```

972 \global\advance\@donereallinesL\@ne
973 \global\advance\@donetotallinesL\@ne
974 \else
975 \setbox\l@leftbox \hb@xt@ \Lcolwidth{\hspace*\Lcolwidth}}%
976 \global\advance\@donetotallinesL\@ne
977 \fi}
978
979

```

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

```

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

```

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

```

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

```

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

```

\do@lineRhook 1007 \newcommand*{\do@lineLhook}{%
\do@insidelineLhook 1008 \newcommand*{\do@lineRhook}{%
\do@insidelineRhook 1009 \newcommand*{\do@insidelineLhook}{%
1010 \newcommand*{\do@insidelineRhook}{%
1011

```

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

```

1012 \newcommand*{\do@lineR}{%
1013   \ledRcol@true%
1014   \advance\countRline \@ne
1015   \ifvbox\namebox{l@dRcolrawbox\the\l@dpscR}%
1016   {\vbadness=10000
1017    \splittopskip=\z@
1018    \do@lineRhook
1019    \l@emptyd@ta
1020    \global\setbox\one@lineR=\vsplit\namebox{l@dRcolrawbox\the\l@dpscR}
1021                                to\baselineskip}%
1022   \IfStrEq{\splitfirstmarks\parledgroup@}{begin}{\parledgroup@notes@startR}{%
1023   \unvbox\one@lineR \global\setbox\one@lineR=\lastbox
1024   \getline@numR
1025   \ifnum\@lockR>\@ne%
1026     \inserthangingsymbolRtrue
1027   \else%
1028     \inserthangingsymbolRfalse%
1029   \fi%
1030   \setbox\l@drightbox
1031   \hb@xt@ \Rcolwidth{%
1032     \affixline@numR%
1033     \xifinlist{\the\l@dpscR}{\eled@sectionsR@@}%
1034     {}%
1035     {\print@lineR}%
1036   }%
1037   \add@penaltiesR
1038   \global\advance\@donereallinesR\@ne
1039   \global\advance\@donetotallinesR\@ne
1040 \else
1041   \setbox\l@drightbox \hb@xt@ \Rcolwidth{\hspace*{\Rcolwidth}}
1042   \global\advance\@donetotallinesR\@ne
1043 \fi
1044 \ledRcol@false%
1045 }
1046
1047

```

```

\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.

```

1048 \newcommand*{\getline@numR}{%
1049   \global\advance\absline@numR \@ne
1050   \do@actionsR
1051   \do@ballastR
1052   \ifledgroupnotesR\else\ifnumberline
1053     \ifsublines@

```

```

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

```

`\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.

```

1086 \newcommand*{\do@ballastR}{\global\ballast@count=\z@
1087 \begingroup
1088 \advance\absline@numR \@ne
1089 \ifnum\next@actionlineR=\absline@numR
1090 \ifnum\next@actionR>-1001
1091 \global\advance\ballast@count by -\c@ballast
1092 \fi
1093 \fi
1094 \endgroup}

```

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

`\do@actions@fixedcodeR`

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

```

1095 \newcommand*{\do@actions@fixedcodeR}{%
1096   \ifcase\@l@dttempcnta%
1097     \or%                               % 1001
1098       \global\sublines@true
1099     \or%                               % 1002
1100       \global\sublines@false
1101     \or%                               % 1003
1102       \global\@lockR=\@ne
1103     \or%                               % 1004
1104       \ifnum\@lockR=\tw@
1105         \global\@lockR=\thr@@
1106       \else
1107         \global\@lockR=\z@
1108       \fi
1109     \or%                               % 1005
1110       \global\sub@lockR=\@ne
1111     \or%                               % 1006
1112       \ifnum\sub@lockR=\tw@
1113         \global\sub@lockR=\thr@@
1114       \else
1115         \global\sub@lockR=\z@
1116       \fi
1117     \or%                               % 1007
1118       \l@dskipnumbertrue
1119     \else
1120       \led@warn@BadAction
1121   \fi}
1122
1123
1124 \newcommand*{\do@actionsR}{%
1125   \global\let\do@actions@nextR=\relax
1126   \@l@dttempcntb=\absline@numR
1127   \ifnum\@l@dttempcntb<\next@actionlineR\else
1128     \ifnum\next@actionR>-1001\relax
1129       \global\page@numR=\next@actionR
1130       \ifbypage@R
1131         \global\line@numR \z@ \global\subline@numR \z@
1132       \fi
1133     \else
1134       \ifnum\next@actionR<-4999\relax    % 9/05 added relax here
1135         \@l@dttempcnta=-\next@actionR
1136         \advance\@l@dttempcnta by -5001\relax
1137         \ifsublines@
1138           \global\subline@numR=\@l@dttempcnta
1139         \else
1140           \global\line@numR=\@l@dttempcnta
1141         \fi
1142       \else
1143         \@l@dttempcnta=-\next@actionR
1144         \advance\@l@dttempcnta by -1000\relax

```

```

1145     \do@actions@fixedcodeR
1146     \fi
1147 \fi
1148 \ifx\actionlines@listR\empty
1149     \gdef\next@actionlineR{1000000}%
1150 \else
1151     \gl@p\actionlines@listR\to\next@actionlineR
1152     \gl@p\actions@listR\to\next@actionR
1153     \global\let\do@actions@nextR=\do@actionsR
1154 \fi
1155 \fi
1156 \do@actions@nextR}
1157

```

17.4 Line number printing

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

```

\ch@cksub@l@ckR 1158
\ch@ck@l@ckR 1159 \providecommand*\l@dcalcnnum}[3]{%
\fx@l@cksR 1160 \ifnum #1 > #2\relax
\affixline@numR 1161 \l@dtempcnta = #1\relax
1162 \advance\l@dtempcnta by -#2\relax
1163 \divide\l@dtempcnta by #3\relax
1164 \multiply\l@dtempcnta by #3\relax
1165 \advance\l@dtempcnta by #2\relax
1166 \else
1167 \l@dtempcnta=#2\relax
1168 \fi}
1169
1170 \newcommand*\ch@cksub@l@ckR}{%
1171 \ifcase\sub@lockR
1172 \or
1173 \ifnum\sublock@disp=\@ne
1174 \l@dtempcntb \z@ \l@dtempcnta \@ne
1175 \fi
1176 \or
1177 \ifnum\sublock@disp=\tw@
1178 \else
1179 \l@dtempcntb \z@ \l@dtempcnta \@ne
1180 \fi
1181 \or
1182 \ifnum\sublock@disp=\z@
1183 \l@dtempcntb \z@ \l@dtempcnta \@ne
1184 \fi
1185 \fi}
1186
1187 \newcommand*\ch@ck@l@ckR}{%
1188 \ifcase\@lockR
1189 \or

```

```

1190 \ifnum\lock@disp=\@ne
1191 \l@dttempcntb \z@ \l@dttempcnta \@ne
1192 \fi
1193 \or
1194 \ifnum\lock@disp=\tw@
1195 \else
1196 \l@dttempcntb \z@ \l@dttempcnta \@ne
1197 \fi
1198 \or
1199 \ifnum\lock@disp=\z@
1200 \l@dttempcntb \z@ \l@dttempcnta \@ne
1201 \fi
1202 \fi}
1203
1204 \newcommand*{\f@x@l@cksR}{%
1205 \ifcase\@lockR
1206 \or
1207 \global\@lockR \tw@
1208 \or \or
1209 \global\@lockR \z@
1210 \fi
1211 \ifcase\sub@lockR
1212 \or
1213 \global\sub@lockR \tw@
1214 \or \or
1215 \global\sub@lockR \z@
1216 \fi}
1217
1218
1219 \newcommand*{\affixline@numR}{%
1220 \ifledgroupnotesR\else\ifnumberline
1221 \ifl@dskipnumber
1222 \global\l@dskipnumberfalse
1223 \else
1224 \ifsublines@
1225 \l@dttempcntb=\subline@numR
1226 \l@dcalcnm{\subline@numR}{\c@firstsublinenumR}{\c@sublinenumincrementR}%
1227 \ch@cksub@lockR
1228 \else
1229 \l@dttempcntb=\line@numR
1230 \ifx\linenumberlist\empty
1231 \l@dcalcnm{\line@numR}{\c@firstlinenumR}{\c@linenumincrementR}%
1232 \else
1233 \l@dttempcnta=\line@numR
1234 \edef\rem@inder{\linenumberlist,\number\line@numR,%
1235 \edef\sc@n@list{\def\noexpand\sc@n@list
1236 ###1,\number\l@dttempcnta,###2|{\def\noexpand\rem@inder{###2}}}%
1237 \sc@n@list\expandafter\sc@n@list\rem@inder|
1238 \ifx\rem@inder\empty\advance\l@dttempcnta\@ne\fi
1239 \fi

```



```

1240     \ch@ck@l@ckR
1241   \fi
1242   \ifnum\@l@dttempcnta=\@l@dttempcntb
1243     \if@twocolumn
1244       \if@firstcolumn
1245         \gdef\l@dld@ta{\llap{\leftlinenumR}}}%
1246       \else
1247         \gdef\l@drd@ta{\rlap{\rightlinenumR}}}%
1248       \fi
1249     \else
1250       \@l@dttempcntb=\line@marginR
1251       \ifnum\@l@dttempcntb>\@ne
1252         \advance\@l@dttempcntb by\page@numR
1253       \fi
1254       \ifodd\@l@dttempcntb
1255         \gdef\l@drd@ta{\rlap{\rightlinenumR}}}%
1256       \else
1257         \gdef\l@dld@ta{\llap{\leftlinenumR}}}%
1258       \fi
1259     \fi
1260   \fi
1261   \f@x@l@cksR
1262 \fi
1263 \fi
1264 \fi}

```

17.5 Pstart number printing in side

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

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

```

\affixpstart@numL
\affixpstart@numR 1265
  \leftpstartnumR 1266 \newcommand*{\affixpstart@numL}{%
\rightpstartnumR 1267 \ifsidepstartnum
\leftpstartnumL 1268 \if@twocolumn
\rightpstartnumL 1269   \if@firstcolumn
  \ifpstartnumR 1270     \gdef\l@dld@ta{\llap{\leftpstartnumL}}}%
    1271   \else
    1272     \gdef\l@drd@ta{\rlap{\rightpstartnumL}}}%
    1273   \fi
    1274   \else
    1275     \@l@dttempcntb=\line@margin
    1276     \ifnum\@l@dttempcntb>\@ne
    1277       \advance\@l@dttempcntb \page@num

```

```

1278 \fi
1279 \ifodd\@l@tempcntb
1280 \gdef\l@drd@ta{\rlap{\rightpstartnumL}}}%
1281 \else
1282 \gdef\l@dld@ta{\llap{\leftpstartnumL}}}%
1283 \fi
1284 \fi
1285 \fi
1286 }
1287 \newcommand*{\affixpstart@numR}{%
1288 \ifsidepstartnum
1289 \if@twocolumn
1290 \if@firstcolumn
1291 \gdef\l@dld@ta{\llap{\leftpstartnumR}}}%
1292 \else
1293 \gdef\l@drd@ta{\rlap{\rightpstartnumR}}}%
1294 \fi
1295 \else
1296 \@l@tempcntb=\line@marginR
1297 \ifnum\@l@tempcntb>\@ne
1298 \advance\@l@tempcntb \page@numR
1299 \fi
1300 \ifodd\@l@tempcntb
1301 \gdef\l@drd@ta{\rlap{\rightpstartnumR}}}%
1302 \else
1303 \gdef\l@dld@ta{\llap{\leftpstartnumR}}}%
1304 \fi
1305 \fi
1306 \fi
1307 }
1308
1309 \newcommand*{\leftpstartnumL}{
1310 \ifpstartnum
1311 \thepstartL
1312 \kern\linenumsep\global\pstartnumfalse\fi
1313 }
1314 \newcommand*{\rightpstartnumL}{
1315 \ifpstartnum\kern\linenumsep
1316 \thepstartL
1317 \global\pstartnumfalse\fi
1318 }
1319 \newif\ifpstartnumR
1320 \pstartnumRtrue
1321 \newcommand*{\leftpstartnumR}{
1322 \ifpstartnumR
1323 \thepstartR
1324 \kern\linenumsep\global\pstartnumRfalse\fi
1325 }
1326 \newcommand*{\rightpstartnumR}{
1327 \ifpstartnumR\kern\linenumsep

```

```

1328 \thepstartR
1329 \global\pstartnumRfalse\fi
1330 }

```

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.

```

1331 \list@create{\inserts@listR}

```

`\add@insertsR` The right text version.

```

\add@inserts@nextR 1332 \newcommand*{\add@insertsR}{%
1333   \global\let\add@inserts@nextR=\relax
1334   \ifx\inserts@listR\empty \else
1335     \ifx\next@insertR\empty
1336       \ifx\insertlines@listR\empty
1337         \global\noteschanged@true
1338         \gdef\next@insertR{100000}%
1339       \else
1340         \gl@p\insertlines@listR\to\next@insertR
1341       \fi
1342     \fi
1343     \ifnum\next@insertR=\absline@numR
1344       \gl@p\inserts@listR\to\@insertR
1345       \@insertR
1346       \global\let\@insertR=\undefined
1347       \global\let\next@insertR=\empty
1348       \global\let\add@inserts@nextR=\add@insertsR
1349     \fi
1350   \fi
1351   \add@inserts@nextR}
1352

```

17.7 Penalties

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

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

```

\newcommand*{\add@penaltiesR}{\@l@tempcnta=\ballast@count

```

```

\ifnum\num@linesR>\@ne
  \global\advance\par@lineR \@ne
  \ifnum\par@lineR=\@ne
    \advance\@l@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.

```

1353 \newcommand*{\add@penaltiesL}{%
1354 \newcommand*{\add@penaltiesR}{%
1355

```

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.

```

1356 \newcommand*{\flush@notesR}{%
1357   \@xloop
1358   \ifx\inserts@listR\empty \else
1359     \gl@p\inserts@listR\to\@insertR
1360     \@insertR
1361     \global\let\@insertR=\undefined
1362   \repeat}
1363

```

18 Footnotes

18.1 Normal footnote formatting

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

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

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

```

1364 \def\printlinesR#1|#2|#3|#4|#5|#6|#7|{\begingroup
1365   \setprintlines{#1}{#2}{#3}{#4}{#5}{#6}%
1366   \ifl@d@pnum #1\fullstop\fi
1367   \ifledplinenum \linenumr@p{#2}\Rlineflag\else \symlinenum\fi
1368   \ifl@d@ssub \fullstop \sublinenumr@p{#3}\fi
1369   \ifl@d@dash \endashchar\fi
1370   \ifl@d@pnum #4\fullstop\fi
1371   \ifl@d@elin \linenumr@p{#5}\Rlineflag\fi
1372   \ifl@d@esl \ifl@d@elin \fullstop\fi \sublinenumr@p{#6}\fi
1373 \endgroup}
1374
1375 \let\ledsavedprintlines\printlines
1376
```

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.

```

1377 \list@create{\labelref@listR}
1378
```

`\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.

```

1379 \renewcommand*{\edlabel}[1]{\@bsphack
1380   \ifledRcol
1381     \write\linenum@outR{\string\@lab}%
1382     \ifx\labelref@listR\empty
1383       \xdef\label@refs{\zz@@@}%
1384     \else
```

```

1385     \gl@p\labelref@listR\to\label@refs
1386     \fi
1387     \ifvmode
1388         \advancelabel@refs
1389     \fi
1390     \protected@write\@auxout{%
1391         {\string\l@dmake@labelsR\space\thepage|\label@refs|\the\c@pstartR|{#1}}%
1392     \else
1393         \write\linenum@out{\string\@lab}%
1394         \ifx\labelref@list\empty
1395             \xdef\label@refs{\zz@@}%
1396         \else
1397             \gl@p\labelref@list\to\label@refs
1398         \fi
1399         \ifvmode
1400             \advancelabel@refs
1401         \fi
1402         \protected@write\@auxout{%
1403             {\string\l@dmake@labelsR\space\thepage|\label@refs|\the\c@pstartR|{#1}}%
1404         \fi
1405         \@esphack}
1406
1407

```

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

```

1408 \def\l@dmake@labelsR#1|#2|#3|#4|#5{%
1409     \expandafter\ifx\csname the@label#5\endcsname \relax\else
1410         \led@warn@DuplicateLabel{#4}%
1411     \fi
1412     \expandafter\gdef\csname the@label#5\endcsname{#1|#2\Rlineflag|#3|#4}%
1413     \ignorespaces}
1414 \AtBeginDocument{%
1415     \def\l@dmake@labelsR#1|#2|#3|#4|#5{%
1416 }
1417

```

`\@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.

```

1418 \renewcommand*{\@lab}{%
1419     \ifledRcol
1420         \xright@appenditem{\linenumr@p{\line@numR}}{|%
1421             \ifsublines@ \sublinenumr@p{\subline@numR}\else 0\fi}%
1422         \to\labelref@listR
1423     \else
1424         \xright@appenditem{\linenumr@p{\line@num}}{|%
1425             \ifsublines@ \sublinenumr@p{\subline@num}\else 0\fi}%
1426         \to\labelref@list

```

```

1427 \fi}
1428

```

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* 1429 \WithSuffix\newcommand\sidenotemargin*[1]{%
1430 \l@dgetsidenote@margin{#1}
1431 \global\sidenote@marginR=\@l@dttempcntb
1432 \global\sidenote@margin=\@l@dttempcntb
1433 }
1434 \newcount\sidenote@marginR
1435 \global\sidenote@margin=\@ne
1436

```

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

```

1437 \newcommand*\affixside@noteR{%
1438 \def\sidenotecontent@{}%
1439 \numgdef\itemcount@{0}%
1440 \def\do##1{%
1441 \ifnumequal{\itemcount@}{0}%
1442 {%
1443 \appto\sidenotecontent@{##1}}% Not print not separator before the 1st note
1444 {\appto\sidenotecontent@{\sidenotesep ##1}}%
1445 }%
1446 \numgdef\itemcount@{\itemcount@+1}%
1447 }%
1448 \dolistloop{\l@dcnotesnotetext}%
1449 \ifnumgreater{\itemcount@}{1}{\led@err@ManySidenotes}{}%
1450 \gdef\@templ@d{%
1451 \gdef\@templ@n{\l@dcnotesnotetext\l@dcnotesnotetext@l\l@dcnotesnotetext@r}%
1452 \ifx\@templ@d\@templ@n \else%
1453 \if@twocolumn%
1454 \if@firstcolumn%
1455 \setl@dlp@rbox{##1}{\sidenotecontent@}%
1456 \else%
1457 \setl@drp@rbox{\sidenotecontent@}%
1458 \fi%
1459 \else%
1460 \@l@dttempcntb=\sidenote@marginR%
1461 \ifnum\@l@dttempcntb>\@ne%
1462 \advance\@l@dttempcntb by\page@numR%
1463 \fi%
1464 \ifodd\@l@dttempcntb%
1465 \setl@drp@rbox{\sidenotecontent@}%

```

```

1466     \gdef\sidenotecontent@{}%
1467     \numdef{\itemcount@}{0}%
1468     \dolistloop{\l@dcstotetext@l}%
1469     \ifnumgreater{\itemcount@}{1}{\led@err@ManyLeftnotes}{}%
1470     \setl@dlp@rbox{\sidenotecontent@}%
1471     \else%
1472     \setl@dlp@rbox{\sidenotecontent@}%
1473     \gdef\sidenotecontent@{}%
1474     \numdef{\itemcount@}{0}%
1475     \dolistloop{\l@dcstotetext@r}%
1476     \ifnumgreater{\itemcount@}{1}{\led@err@ManyRightnotes}{}%
1477     \setl@drp@rbox{\sidenotecontent@}%
1478     \fi%
1479   \fi%
1480 \fi%
1481 }
1482

```

21 Familiar footnotes

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

```

1483 \renewcommand{\l@dbfnote}[1]{%
1484   \ifnumberedpar@
1485   \gdef\@tag{#1}%
1486   \ifledRcol%
1487     \xright@appenditem{\noexpand\vl@dbfnote{\csexpandonce{\@tag}}{\@thefnmark}}%
1488     \to\inserts@listR
1489     \global\advance\insert@countR \one%
1490   \else%
1491     \xright@appenditem{\noexpand\vl@dbfnote{\csexpandonce{\@tag}}{\@thefnmark}}%
1492     \to\inserts@list
1493     \global\advance\insert@count \one%
1494   \fi
1495 \fi\ignorespaces}
1496

```

`\normalbfnoteX`

```

1497 \renewcommand{\normalbfnoteX}[2]{%
1498   \ifnumberedpar@
1499   \ifledRcol%
1500     \ifluatex
1501       \footnotelang@lua[R]%
1502     \fi
1503     \@ifundefined{xpg@main@language}%if polyglossia
1504     {}%
1505     {\footnotelang@poly[R]}%
1506   \protected@csxdef{thisfootnote}{\csuse{thefootnote#1}}%

```



```

1507     \xright@appenditem{\noexpand\vbfnoteX{#1}{#2}{\csexpandonce{thisfootnote}}}%
1508                     \to\inserts@listR
1509     \global\advance\insert@countR \@ne%
1510 \else%
1511     \ifluatex
1512         \footnotelang@lua%
1513     \fi
1514     \@ifundefined{xpg@main@language}%if polyglossia
1515     {}%
1516     {\footnotelang@poly}%
1517     \protected@csxdef{thisfootnote}{\csuse{thefootnote#1}}%
1518     \xright@appenditem{\noexpand\vbfnoteX{#1}{#2}{\csexpandonce{thisfootnote}}}%
1519                     \to\inserts@list
1520     \global\advance\insert@count \@ne%
1521 \fi
1522 \fi\ignorespaces}
1523

```

22 Verse

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

```

\inserthangingsymbolL
\inserthangingsymbolR 1524 \newif\ifinserthangingsymbolR
1525 \newcommand{\inserthangingsymbolL}{%
1526 \ifinserthangingsymbol%
1527     \ifinstanzaL%
1528         \hangingsymbol%
1529     \fi%
1530 \fi}
1531 \newcommand{\inserthangingsymbolR}{%
1532 \ifinserthangingsymbolR%
1533     \ifinstanzaR%
1534         \hangingsymbol%
1535     \fi%
1536 \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 1537 \newcommand{\correcthangingL}{%
1538 \ifl@dpageing\else%
1539     \ifinstanzaL%
1540         \ifinserthangingsymbol%
1541             \hskip \@ifundefined{sza@00}{0}{\expandafter%

```

```

1542          \noexpand\csname sza@0@\endcsname\stanzaindentbase%
1543      \fi%
1544  \fi%
1545 \fi}
1546
1547 \newcommand{\correcthangingR}{%
1548 \ifl@dpaging\else%
1549     \ifinstanzaR%
1550         \ifinserthangingsymbolR%
1551             \hskip \@ifundefined{sza@0@}{0}{\expandafter%
1552                 \noexpand\csname sza@0@\endcsname\stanzaindentbase%
1553             \fi%
1554         \fi%
1555     \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.

```

1556 \chardef\next=\catcode'\&
1557 \catcode'\&=\active
1558

```

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

```

1559 \newenvironment{astanza}{%
1560     \startstanzahook
1561     \catcode'\&\active
1562     \global\stanza@count\@ne\stanza@modulo\@ne
1563     \ifnum\usernamecount{sza@0@}=\z@
1564         \let\stanza@hang\relax
1565     \let\endlock\relax
1566     \else
1567     %% \interlinepenalty\@M % this screws things up, but I don't know why
1568     \rightskip\z@ plus 1fil\relax
1569     \fi
1570     \ifnum\usernamecount{szp@0@}=\z@
1571         \let\sza@penalty\relax
1572     \fi
1573     \def&{%
1574         \endlock\mbox{}%
1575         \sza@penalty
1576         \global\advance\stanza@count\@ne
1577         \@astanza@line}%
1578     \def\&{%
1579         \endlock\mbox{}
1580         \pend
1581         \endstanzaextra}%
1582     \pstart
1583     \@astanza@line

```

```

1584 }{}
1585

```

`\@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.

```

1586 \newcommand*{\@astanza@line}{%
1587   \ifnum\value{stanzaindentrepetition}=0
1588     \parindent=\csname sza@\number\stanza@count
1589             @\endcsname\stanzaindentbase
1590   \else
1591     \parindent=\csname sza@\number\stanza@modulo
1592             @\endcsname\stanzaindentbase
1593     \managestanza@modulo
1594   \fi
1595   \par
1596   \stanza@hang%\mbox{}%
1597   \ignorespaces}
1598

```

Lastly reset the modified category codes.

```

1599 \catcode'\&=\next
1600

```

23 Naming macros

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

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

```

\unhnamebox 1601 \providecommand*{\newnamebox}[1]{%
\unvnamebox 1602   \expandafter\newbox\csname #1\endcsname}
\namebox 1603 \providecommand*{\setnamebox}[1]{%
1604   \expandafter\setbox\csname #1\endcsname}
1605 \providecommand*{\unhnamebox}[1]{%
1606   \expandafter\unhbox\csname #1\endcsname}
1607 \providecommand*{\unvnamebox}[1]{%
1608   \expandafter\unvbox\csname #1\endcsname}
1609 \providecommand*{\namebox}[1]{%
1610   \csname #1\endcsname}
1611

```

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

```

\usenamecount 1612 \providecommand*{\newnamecount}[1]{%
1613   \expandafter\newcount\csname #1\endcsname}
1614 \providecommand*{\usenamecount}[1]{%
1615   \csname #1\endcsname}
1616

```

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.
1617 \newcount\l@dc@maxchunks
1618 \newcommand{\maxchunks}[1]{\l@dc@maxchunks=#1}
1619 \maxchunks{5120}
1620

\l@dnumpstartsL The numbers of left and right chunks. \l@dnumpstartsL is defined in eledmac.
\l@dnumpstartsR 1621 \newcount\l@dnumpstartsR
1622

\l@pscL A couple of scratch counts for use in left and right texts, respectively.
\l@pscR 1623 \newcount\l@dpscL
1624 \newcount\l@dpscR
1625

\l@dsetuprawboxes This macro creates \maxchunks pairs of boxes for left and right chunks. The boxes
are called \l@dLcolrawbox1, \l@dLcolrawbox2, etc.
1626 \newcommand*{\l@dsetuprawboxes}{%
1627 \l@l@tempcntb=\l@dc@maxchunks
1628 \loop\ifnum\l@l@tempcntb>\z@
1629 \newnamebox{\l@dLcolrawbox\the\l@l@tempcntb}
1630 \newnamebox{\l@dRcolrawbox\the\l@l@tempcntb}
1631 \advance\l@l@tempcntb \m@ne
1632 \repeat}
1633

\l@dsetupmaxlinecounts To be able to synchronise left and right texts we need to know the maximum num-
\l@dzeromaxlinecounts ber of text lines there are in each pair of chunks. \l@dsetupmaxlinecounts creates
\maxchunks new counts called \l@dmaxlinesinpar1, etc., and \l@dzeromaxlinecounts
zeroes all of them.
1634 \newcommand*{\l@dsetupmaxlinecounts}{%
1635 \l@l@tempcntb=\l@dc@maxchunks
1636 \loop\ifnum\l@l@tempcntb>\z@
1637 \newnamecount{\l@dmaxlinesinpar\the\l@l@tempcntb}
1638 \advance\l@l@tempcntb \m@ne
1639 \repeat}
1640 \newcommand*{\l@dzeromaxlinecounts}{%
1641 \begingroup
1642 \l@l@tempcntb=\l@dc@maxchunks
1643 \loop\ifnum\l@l@tempcntb>\z@

```

```

1644 \global\usernamecount{l@maxlinesinpar\the\l@dttempcntb}=\z@
1645 \advance\l@dttempcntb \m@ne
1646 \repeat
1647 \endgroup}
1648

```

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

```

1649 \AtBeginDocument{%
1650 \l@dsetuprawboxes
1651 \l@dsetupmaxlinecounts
1652 \l@dzeromaxlinecounts
1653 \l@dnumstartsl=\z@
1654 \l@dnumstartsr=\z@
1655 \l@dpscl=\z@
1656 \l@dpscr=\z@}
1657

```

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 1658 \newif\ifl@dusedbabel
\l@dusedbabeltrue 1659 \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{\langle lang\rangle} does the work when the language
\langle lang\rangle 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 \.

1660 \newcommand*{\l@dbbl@set@language}[1]{%
1661 \edef\language{#1}%
1662 \select@language{\language}%

```

```

1663 \if@filesw
1664   \protected@write\@auxout{}\string\select@language{\language}%
1665   \addtocontents{toc}{\string\select@language{\language}}%
1666   \addtocontents{lof}{\string\select@language{\language}}%
1667   \addtocontents{lot}{\string\select@language{\language}}%
1668 \fi}
1669

```

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
\l@duselanguage are the names of the languages of the left and right texts. \l@duselanguage is
\theledlanguageL similar to \selectlanguage.
\theledlanguageR
1670 \providecommand{\selectlanguage}[1]{%
1671 \newcommand*\l@duselanguage[1]{%
1672 \gdef\theledlanguageL{}
1673 \gdef\theledlanguageR{}
1674

```

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

```

1675 \AtBeginDocument{%
1676   \ifundefined{xpg@main@language}{%
1677     \ifundefined{bbl@main@language}{%

```

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

```

1678   \l@dusedbabelfalse
1679   \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.

```

1680   \l@dusedbabeltrue
1681   \let\l@doldselectlanguage\selectlanguage
1682   \let\l@doldbbl@set@language\bbl@set@language
1683   \let\bbl@set@language\l@dbbl@set@language
1684   \renewcommand{\selectlanguage}[1]{%
1685     \l@doldselectlanguage{#1}%
1686     \ifledRcol \gdef\theledlanguageR{#1}%
1687     \else      \gdef\theledlanguageL{#1}%
1688   \fi}

```

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

```

1689   \renewcommand*\l@duselanguage[1]{%
1690     \l@doldselectlanguage{#1}}

```

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

```

1691   \gdef\theledlanguageL{\bbl@main@language}%
1692   \gdef\theledlanguageR{\bbl@main@language}%

```

```

1693 }%
1694 }
    If on Polyglossia
1695 { \let\old@otherlanguage\otherlanguage%
1696   \renewcommand{\otherlanguage}[2] [] {%
1697     \selectlanguage[#1]{#2}%
1698     \ifledRcol \gdef\theledlanguageR{#2}%
1699     \else      \gdef\theledlanguageL{#2}%
1700     \fi}%
1701   \let\l@duselanguage\xpg@set@language%
1702   \gdef\theledlanguageL{\xpg@main@language}%
1703   \gdef\theledlanguageR{\xpg@main@language}%
1704 % \end{macrocode}
1705 % That's it.
1706 % \begin{macrocode}
1707 }}

```

26 Parallel columns

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

```

\@eledsectionR 1708 \newsavebox{\@eledsectionL}%
1709 \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.

```

1710 \newcommand*{\Columns}{%
1711   \eledsection@correcting@skip=-\baselineskip% Correction for sections' titles
1712   \setcounter{pstartL}{\value{pstartLold}}
1713   \setcounter{pstartR}{\value{pstartRold}}
1714   \ifnum\l@dnumpstartsL=\l@dnumpstartsR\else
1715     \led@err@BadLeftRightPstarts{\the\l@dnumpstartsL}{\the\l@dnumpstartsR}%
1716   \fi

```

Start a group and zero counters, etc.

```

1717 \begingroup
1718 \l@dzeropenalties
1719 \endgraf\global\num@lines=\prevgraf
1720 \global\num@linesR=\prevgraf
1721 \global\par@line=\z@
1722 \global\par@lineR=\z@
1723 \global\l@dpscL=\z@
1724 \global\l@dpscR=\z@

```

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

```

1725 \check@pstarts
1726 \loop\if@pstarts

```

```

1727      \global\pstartnumtrue
1728      \global\pstartnumRtrue

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

1729      \global\advance\l@dpscL \@ne
1730      \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

1731      \checkraw@text
1732 {      \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.

1733      \l@duselanguage{\theledlanguageL}%
1734      \do@lineL
1735      \xifinlist{\the\l@dpscL}{\eled@sections@@}
1736      {%
1737      \ifdefstring{\@eledsectmark}{L}%
1738      {\csuse{eled@sectmark@the\l@dpscL}%
1739      }-}%
1740      \global\csundef{eled@sectmark@the\l@dpscL}%
1741      \savebox{\@eledsectionL}{\parbox[t][t]{\Lcolwidth}{\vbox{} \print@eledse
1742      }%
1743      }%
1744      \l@duselanguage{\theledlanguageR}%
1745      \do@lineR
1746      \xifinlist{\the\l@dpscR}{\eled@sectionsR@@}
1747      {%
1748      \ifdefstring{\@eledsectmark}{R}%
1749      {\csuse{eled@sectmark@the\l@dpscR R}%
1750      }-}%
1751      \global\csundef{eled@sectmark@the\l@dpscR R}%
1752      \savebox{\@eledsectionR}{\parbox[t][t]{\Rcolwidth}{\vbox{} \print@eledse
1753      }%
1754      \hb@xt@ \hsize{%
1755      \ifdefstring{\columns@position}{L}{-}{\hfill }%
1756      \unhbox\l@dleftbox%
1757      \ifhbox\@eledsectionL%
1758      \usebox{\@eledsectionL}%
1759      \fi%
1760      \print@columnseparator%
1761      \unhbox\l@drightbox%
1762      \ifhbox\@eledsectionR%
1763      \usebox{\@eledsectionR}%
1764      \fi%
1765      \ifdefstring{\columns@position}{R}{-}{\hfill}%
1766      }%
1767      \checkraw@text
1768      \checkverseL

```



```

1769         \checkverseR
1770         \checkpb@columns
1771     \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.

```

1772     \@writelinesinparL
1773     \@writelinesinparR
1774     \check@pstarts
1775         \ifbypstart@
1776             \write\linenum@out{\string\@set[1]}
1777             \resetprevline@
1778         \fi
1779         \ifbypstart@R
1780             \write\linenum@outR{\string\@set[1]}
1781             \resetprevline@
1782         \fi
1783     \addtocounter{pstartL}{1}
1784     \addtocounter{pstartR}{1}
1785 \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.

```

1786     \flush@notes
1787     \flush@notesR
1788 \endgroup
1789 \global\l@dpscL=\z@
1790 \global\l@dpscR=\z@
1791 \global\l@dnumpstartsL=\z@
1792 \global\l@dnumpstartsR=\z@
1793 \ignorespaces
1794 \global\instanzaLfalse
1795 \global\instanzaRfalse}
1796

```

`\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).

```

1797 \def\print@columnseparator{%
1798     \ifdim\beforecolumnseparator<0pt%
1799         \hfill%
1800     \else%
1801         \hspace{\beforecolumnseparator}%
1802     \fi%
1803     \columnseparator%
1804     \ifdim\aftercolumnseparator<0pt%
1805         \hfill%
1806     \else%
1807         \hspace{\aftercolumnseparator}%

```

```

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

```

\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).

```

1836 \newcommand*{\columnseparator}{%
1837     \smash{\rule[-0.2\baselineskip]{\columnrulewidth}{1.05\baselineskip}}}
1838 \newdimen\columnrulewidth
1839 \columnrulewidth=\z@
1840

```

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

```

1841 \newcommand*{\columnspostion}[1]{%
1842     \xdef\columns@position{#1}%
1843 }%
1844 \xdef\columns@position{R}%

```

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

```

1845 \newlength{\beforecolumnseparator}%
1846 \setlength{\beforecolumnseparator}{-2pt}%
1847
1848 \newlength{\aftercolumnseparator}%
1849 \setlength{\aftercolumnseparator}{-2pt}%

```

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

```

\@pstartstrue 1850 \newif\if@pstarts
\@pstartsfalse 1851 \newcommand*{\check@pstarts}{%
\check@pstarts 1852 \@pstartsfalse
1853 \ifnum\l@dnumpstartsL>\l@dpscL
1854 \@pstartstrue
1855 \else
1856 \ifnum\l@dnumpstartsR>\l@dpscR
1857 \@pstartstrue
1858 \fi
1859 \fi
1860 }
1861

```

\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 1862 \newif\ifaraw@text
1863 \araw@textfalse
1864 \newcommand*{\checkraw@text}{%
1865 \araw@textfalse
1866 \ifvbox\namebox{\l@dLcolrawbox\the\l@dpscL}
1867 \araw@texttrue
1868 \else
1869 \ifvbox\namebox{\l@dRcolrawbox\the\l@dpscR}
1870 \araw@texttrue
1871 \fi
1872 \fi
1873 }
1874

```

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

```

1875 \newcommand*{\@writelinesinparL}{%
1876 \edef\next{%
1877 \write\linenum@out{\string\@pend[\the\@donereallinesL]}}%
1878 \next
1879 \global\@donereallinesL \z@}
1880 \newcommand*{\@writelinesinparR}{%
1881 \edef\next{%
1882 \write\linenum@outR{\string\@pendR[\the\@donereallinesR]}}%
1883 \next
1884 \global\@donereallinesR \z@}
1885

```

27 Parallel pages

This is considerably more complicated than parallel columns.

```

\l@minpagelinesL Counts for the number of lines on a left or right page, and the smaller of the
\l@minpagelinesR number of lines on a pair of facing pages.
1886 \newcount\l@minpagelinesL
1887 \newcount\l@minpagelinesR
1888 \newcount\l@minpagelines
1889

\Pages The \Pages command results in the previous Left and Right texts being typeset
on matching facing pages. There should be equal numbers of chunks in the left
and right texts.

1890 \newcommand*{\Pages}{%
1891   \eledsection@correcting@skip=-2\baselineskip% line correcting for section titles.
1892   \setcounter{pstartL}{\value{pstartLold}}
1893   \setcounter{pstartR}{\value{pstartRold}}
1894   \parledgroup@notespacing@set@correction
1895   \typeout{}
1896   \typeout{***** PAGES *****}
1897   \ifnum\l@dnumpstartsL=\l@dnumpstartsR\else
1898     \led@err@BadLeftRightPstarts{\the\l@dnumpstartsL}{\the\l@dnumpstartsR}%
1899   \fi

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

1900   \cleartol@devenpage
1901   \begingroup
1902     \l@dzeropenalties
1903     \endgraf\global\l@num@lines=\prevgraf
1904     \global\l@num@linesR=\prevgraf
1905     \global\l@par@line=\z@
1906     \global\l@par@lineR=\z@
1907     \global\l@dpscL=\z@
1908     \global\l@dpscR=\z@
1909     \writtenlinesLfalse
1910     \writtenlinesRfalse

    Check if there are chunks to be processed.

1911     \check@pstarts
1912     \loop\if@pstarts

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

1913       \global\advance\l@dpscL \@ne
1914       \global\advance\l@dpscR \@ne

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

1915       \getlinesfromparlistL

```

```

1916      \getlinesfromparlistR
1917      \l@dcalc@maxoftwo{\@cs@linesinparL}{\@cs@linesinparR}%
1918      {\usernamecount{l@dmaxlinesinpar\the\l@dpscL}}%
1919      \check@pstarts
1920      \repeat

```

Zero the counts again, ready for the next bit.

```

1921      \global\l@dpscL=\z@
1922      \global\l@dpscR=\z@

```

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

```

1923      \getlinesfrompagelistL
1924      \getlinesfrompagelistR
1925      \l@dcalc@minoftwo{\@cs@linesonpageL}{\@cs@linesonpageR}%
1926      {\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.

```

1927      \check@pstarts
1928      \if@pstarts

```

Increment the chunk counts to get the first pair.

```

1929      \global\advance\l@dpscL \@ne
1930      \global\advance\l@dpscR \@ne

```

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

```

1931      \global\@donereallinesL=\z@
1932      \global\@donetotallinesL=\z@
1933      \global\@donereallinesR=\z@
1934      \global\@donetotallinesR=\z@

```

Start a loop over the boxes (chunks).

```

1935      \checkraw@text
1936 %      \begingroup
1937 {      \loop\ifraw@text

```

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

```

1938      \checkpageL
1939      \l@duselanguage{\theledlanguageL}%
1940 %%%      \begingroup
1941 {      \loop\ifl@dsamepage%

```

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

```

1942      \ifdefstring{\@eledsectnotoc}{L}{\ledsectnotoc}{\}
1943      \csuse{before@pstartL@the\l@dpscL}
1944      \global\csundef{before@pstartL@the\l@dpscL}
1945      \do@lineL
1946      \xifinlist{\the\l@dpscL}{\eled@sections@@}

```

```

1947      {\print@eledsectionL}%
1948      {}%
1949      \advance\numpagelinesL \@ne
1950      \ifshiftedpstarts
1951          \ifdim\ht\l@dleftbox>0pt\hb@xt@ \hsize{\ledstrutL\unhbox\l@dleftbox}%
1952      \else%
1953          \parledgroup@correction@notespacing{L}
1954          \hb@xt@ \hsize{\ledstrutL\unhbox\l@dleftbox}%
1955      \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.

```

1956      \get@nextboxL
1957      \checkpageL
1958      \checkverseL
1959      \checkpbL
1960      \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.

```

1961      \ifl@dpagfull
1962          \@writelinesonpageL{\the\numpagelinesL}%
1963      \else
1964          \@writelinesonpageL{1000}%
1965      \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.

```

1966      \numpagelinesL \z@
1967      \parledgroup@correction@notespacing@init
1968      \clearl@dleftpage }%

```

Now do the same for the right text.

```

1969      \checkpageR
1970      \l@duselanguage{\theledlanguageR}%
1971      {
1972          \loop\ifl@dsamepage%
1973              \initnumbering@sectcountR
1974              \ifdefstring{\@eledsectnotoc}{R}{\ledsectnotoc}{%
1975                  \csuse{before@pstartR@the\l@dpscR}
1976                  \global\csundef{before@pstartR@the\l@dpscR}
1977                  \do@lineR
1978                  \xifinlist{\the\l@dpscR}{\eled@sectionsR@@}
1979                  {\print@eledsectionR}%
1980                  {}%
1981              \advance\numpagelinesR \@ne
1982              \ifshiftedpstarts
1983                  \ifdim\ht\l@drighthbox>0pt\hb@xt@ \hsize{\ledstrutR\unhbox\l@drighthbox}%

```

```

1984          \parledgroup@correction@notespacing{R}
1985          \hb@xt@ \hsize{\ledstrutR\unhbox\l@drightbox}%
1986      \fi
1987      \get@nextboxR
1988      \checkpageR
1989      \checkverseR
1990      \checkpbr
1991      \repeat
1992      \ifl@dpagfull
1993          \@writelinesonpageR{\the\numpagelinesR}%
1994      \else
1995          \@writelinesonpageR{1000}%
1996      \fi
1997      \numpagelinesR=\z@
1998      \parledgroup@correction@notespacing@init

```

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

```

1999      \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.

```

2000      \checkraw@text
2001      \ifaraw@text
2002          \getlinesfrompagelistL
2003          \getlinesfrompagelistR
2004          \l@dcalc@minoftwo{\@cs@linesonpageL}{\@cs@linesonpageR}%
2005                          {\l@dminpagelines}%
2006      \fi
2007      \repeat}

```

We have now output the text from all the chunks.

```

2008      \fi

```

Make sure that there are no inserts hanging around.

```

2009      \flush@notes
2010      \flush@notesR
2011  \endgroup

```

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

```

2012  \global\l@dpscL=\z@
2013  \global\l@dpscR=\z@
2014  \global\l@dnumpststartsL=\z@
2015  \global\l@dnumpststartsR=\z@
2016  \global\instanzaLfalse
2017  \global\instanzaRfalse
2018  \ignorespaces}
2019

```

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

```

2020 \newcommand*{\ledstrutL}{\strut}
2021 \newcommand*{\ledstrutR}{\strut}
2022

```

`\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.

```

2023 \providecommand{\cleartoevenpage}[1][\@empty]{%
2024   \clearpage
2025   \ifodd\c@page\hbox{ }#1\clearpage\fi}
2026 \newcommand*{\cleartol@devenpage}{%
2027   \ifdim\pagetotal<\topskip% on an empty page
2028   \else
2029     \clearpage
2030   \fi
2031   \ifodd\c@page\hbox{ }\clearpage\fi}
2032 \newcommand*{\clearl@dleftpage}{%
2033   \clearpage
2034   \ifodd\c@page\else
2035     \led@err@LeftOnRightPage
2036     \hbox{ }%
2037     \cleardoublepage
2038   \fi}
2039 \newcommand*{\clearl@drightpage}{%
2040   \clearpage
2041   \ifodd\c@page
2042     \led@err@RightOnLeftPage
2043     \hbox{ }%
2044     \cleartoevenpage
2045   \fi}
2046

```

`\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 2047 \newcommand*{\getlinesfromparlistL}{%
2048   \ifx\linesinpar@listL\empty
2049     \gdef\@cs@linesinparL{0}%
2050   \else
2051     \gl@p\linesinpar@listL\to\@cs@linesinparL
2052   \fi}
2053 \newcommand*{\getlinesfromparlistR}{%
2054   \ifx\linesinpar@listR\empty
2055     \gdef\@cs@linesinparR{0}%
2056   \else
2057     \gl@p\linesinpar@listR\to\@cs@linesinparR
2058   \fi}

```


2059

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

```
\@cs@linesonpageR 2060 \newcommand*\getlinesfrompagelistL{%
2061   \ifx\linesonpage@listL\empty
2062     \gdef\@cs@linesonpageL{1000}%
2063   \else
2064     \glp\linesonpage@listL\to\@cs@linesonpageL
2065   \fi}
2066 \newcommand*\getlinesfrompagelistR{%
2067   \ifx\linesonpage@listR\empty
2068     \gdef\@cs@linesonpageR{1000}%
2069   \else
2070     \glp\linesonpage@listR\to\@cs@linesonpageR
2071   \fi}
2072
```

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

```
2073 \newcommand*\@writelinesonpageL[1]{%
2074   \edef\next{\write\linenum@out{\string\@lopL{#1}}}%
2075   \next}
2076 \newcommand*\@writelinesonpageR[1]{%
2077   \edef\next{\write\linenum@outR{\string\@lopR{#1}}}%
2078   \next}
2079
```

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

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

```
2080 \newcommand*\l@dcalc@maxoftwo[3]{%
2081   \ifnum #2>#1\relax
2082     #3=#2\relax
2083   \else
2084     #3=#1\relax
2085   \fi}
2086 \newcommand*\l@dcalc@minoftwo[3]{%
2087   \ifnum #2<#1\relax
2088     #3=#2\relax
2089   \else
2090     #3=#1\relax
2091   \fi}
2092
```

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

```
\ifl@dpagetrue
\l@dpagetrue
\l@dpagetruefalse
\checkpageL
\checkpageR
```

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

```

2093 \newif\ifl@dsamepage
2094 \l@dsamepagetrue
2095 \newif\ifl@dpagfull
2096
2097 \newcommand*{\checkpageL}{%
2098   \l@dpagfulltrue
2099   \l@dsamepagetrue
2100   \check@goal
2101   \ifdim\pagetotal<\ledthegoal
2102     \ifnum\numpagelinesL<\l@dminpagelines
2103       \else
2104         \l@dsamepagefalse
2105         \l@dpagfullfalse
2106       \fi
2107     \else
2108       \l@dsamepagefalse
2109       \l@dpagfulltrue
2110     \fi}
2111 \newcommand*{\checkpageR}{%
2112   \l@dpagfulltrue
2113   \l@dsamepagetrue
2114   \check@goal
2115   \ifdim\pagetotal<\ledthegoal
2116     \ifnum\numpagelinesR<\l@dminpagelines
2117       \else
2118         \l@dsamepagefalse
2119         \l@dpagfullfalse
2120       \fi
2121     \else
2122       \l@dsamepagefalse
2123       \l@dpagfulltrue
2124     \fi}
2125

```

`\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`.

```

2126 \newcommand{\checkpbL}{
2127   \IfStrEq{\led@pb@setting}{after}{
2128     \xifinlistcs{\the\absline@num}{\l@prev@pb}{\l@dpagfulltrue\l@dsamepagefalse}{
2129       \xifinlistcs{\the\absline@num}{\l@prev@nopb}{\l@dpagfullfalse\l@dsamepagetrue}{
2130     }}
2131   \IfStrEq{\led@pb@setting}{before}{
2132     \numdef{\next@absline}{\the\absline@num+1}
2133     \xifinlistcs{\next@absline}{\l@prev@pb}{\l@dpagfulltrue\l@dsamepagefalse}{

```

```

2134     \xifinlistcs{\next@absline}{\l@prev@nopb}{\l@dpagfullfalse\l@dsamepagetrue}{\l@dpagfulltrue\l@dsamepagetrue}{\l@dsamepagetrue}{\l@dsamepagetrue}}
2135   }{}
2136 }
2137
2138 \newcommand{\checkpbR}{
2139   \IfStrEq{\led@pb@setting}{after}{
2140     \xifinlistcs{\the\absline@numR}{\l@prev@pbR}{\l@dpagfulltrue\l@dsamepagetrue}{\l@dsamepagetrue}{\l@dsamepagetrue}}
2141     \xifinlistcs{\the\absline@numR}{\l@prev@nopbR}{\l@dpagfullfalse\l@dsamepagetrue}{\l@dsamepagetrue}{\l@dsamepagetrue}}
2142   }{}
2143   \IfStrEq{\led@pb@setting}{before}{
2144     \numdef{\next@abslineR}{\the\absline@numR+1}
2145     \xifinlistcs{\next@abslineR}{\l@prev@pbR}{\l@dpagfulltrue\l@dsamepagetrue}{\l@dsamepagetrue}{\l@dsamepagetrue}}
2146     \xifinlistcs{\next@abslineR}{\l@prev@nopbR}{\l@dpagfullfalse\l@dsamepagetrue}{\l@dsamepagetrue}{\l@dsamepagetrue}}
2147   }{}
2148 }

```

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

```

2149 \newcommand{\checkverseL}{
2150   \ifinstanzaL
2151     \iflednopbinverse
2152       \ifinserthangingsymbol
2153         \numgdef{\prev@abslineverse}{\the\absline@num-1}
2154         \IfStrEq{\led@pb@setting}{after}{\lednopbnum{\prev@abslineverse}}{}
2155         \IfStrEq{\led@pb@setting}{before}{\ifnum\numpagelinesL<3\ledpbnum{\prev@abslineverse}\fi}{}
2156       \fi
2157     \fi
2158   \fi
2159 }
2160 \newcommand{\checkverseR}{
2161   \ifinstanzaR
2162     \iflednopbinverse
2163       \ifinserthangingsymbolR
2164         \numgdef{\prev@abslineverse}{\the\absline@numR-1}
2165         \IfStrEq{\led@pb@setting}{after}{\lednopbnumR{\prev@abslineverse}}{}
2166         \IfStrEq{\led@pb@setting}{before}{\ifnum\numpagelinesR<3\ledpbnumR{\prev@abslineverse}\fi}{}
2167       \fi
2168     \fi
2169   \fi
2170 }

```

`\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`.

```

2171 \newdimen\ledthegoal
2172 \ifshiftedpstarts
2173   \newcommand*{\goalfraction}{0.95}
2174 \else
2175   \newcommand*{\goalfraction}{0.9}

```

```

2176 \fi
2177
2178 \newcommand*{\check@goal}{%
2179   \ledthegoal=\goalfraction\pagegoal}
2180

```

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

```

\ifwrittenlinesL 2181 \newif\ifwrittenlinesL
2182 \newif\ifwrittenlinesR
2183

```

\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.

```

2184 \newcommand*{\get@nextboxL}{%
2185   \ifvbox\namebox{\l@dLcolrawbox\the\l@dpscl}% box is not empty

   The current box is not empty; do nothing.

2186   \else%                                     box is empty

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

2187   \ifnum\usenamecount{\l@dmaxlinesinpar\the\l@dpscl}>\@donetotallinesL
2188     \parledgroup@notes@endL
2189   \else

   Sufficient lines have been output.

2190   \ifnum\usenamecount{\l@dmaxlinesinpar\the\l@dpscl}=\@donetotallinesL
2191     \parledgroup@notes@endL
2192   \fi
2193   \ifwrittenlinesL\else

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

2194   \@writelinesinparL
2195   \writtenlinesLtrue
2196   \fi
2197   \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.

2198   \writtenlinesLfalse
2199   \ifbypstart@
2200     \ifnum\value{pstartL}<\value{pstartLold}
2201     \else
2202       \global\line@num=0
2203       \resetprevline@
2204     \fi
2205   \fi

2206 % Add the content of the optional argument of the previous \cs{pend}.
2207 %   \begin{macrocode}
2208   \csuse{after@pendL@\the\l@dpscl}%
2209   \global\csundef{after@pendL@\the\l@dpscl}%

```

```

2210 % \end{macrocode}
2211 % \begin{macrocode}
2212 \addtocounter{pstartL}{1}
2213 \global\pstartnumtrue
2214 \l@dcalcl@maxoftwo{\the\usernamecount{l@dmaxlinesinpar\the\l@dpscL}}%
2215 \the\@donetotallinesL}%
2216 \usernamecount{l@dmaxlinesinpar\the\l@dpscL}}%
2217 \global\@donetotallinesL \z@
2218 \global\advance\l@dpscL \@ne

Add notes of parallel ledgroup.
2219 \parledgroup@notes@endL
2220 \parledgroup@correction@notespadding@final{L}
2221 \else
2222 % Add the content of the optional argument of the last \cs{pend}.
2223 % \begin{macrocode}
2224 \l@dpagefulltrue
2225 \csuse{after@pendL@the\l@dpscL}%
2226 \global\csundef{after@pendL@the\l@dpscL}%
2227 % \end{macrocode}
2228 % \begin{macrocode}
2229 \fi
2230 \fi
2231 \fi}

2232 \newcommand*{\get@nextboxR}{%
2233 \ifvbox\namebox{l@dRcolrawbox\the\l@dpscR}% box is not empty
2234 \else% box is empty
2235 \ifnum\usernamecount{l@dmaxlinesinpar\the\l@dpscR}>\@donetotallinesR
2236 \parledgroup@notes@endR
2237 \else
2238 \ifnum\usernamecount{l@dmaxlinesinpar\the\l@dpscR}=\@donetotallinesR
2239 \parledgroup@notes@endR
2240 \fi
2241 \ifwrittenlinesR\else
2242 \@writelinesinparR
2243 \writtenlinesRtrue
2244 \fi
2245 \ifnum\l@dnumpstartsR>\l@dpscR
2246 \writtenlinesRfalse
2247 \ifbypstart@R
2248 \ifnum\value{pstartR}<\value{pstartRold}
2249 \else
2250 \global\line@numR=0
2251 \resetprevline@
2252 \fi
2253 \fi
2254 \csuse{after@pendR@the\l@dpscR}%
2255 \global\csundef{after@pendR@the\l@dpscR}%
2256 \addtocounter{pstartR}{1}
2257 \global\pstartnumRtrue

```

```

2258      \l@dcalcm@maxoftwo{\the\usernamecount{l@dmaxlinesinpar\the\l@dpscR}}}%
2259      {\the\@donetotallinesR}%
2260      {\usernamecount{l@dmaxlinesinpar\the\l@dpscR}}}%
2261      \global\@donetotallinesR \z@
2262      \global\advance\l@dpscR \@ne
2263      \parledgroup@notes@endR
2264      \parledgroup@correction@notes@spacing@final{R}
2265      \else
2266      \csuse{after@pendR@the\l@dpscR}%
2267      \global\csundef{after@pendR@the\l@dpscR}%
2268      \l@dpagfulltrue
2269      \fi
2270      \fi
2271      \fi}
2272

```

28 Sections' titles' commands

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

```

2273 \newcommand{\eledsectnotoc}[1]{\xdef\@eledsectnotoc{#1}}
2274 \eledsectnotoc{R}

```

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

```

2275 \newcommand{\eledsectmark}[1]{\xdef\@eledsectmark{#1}}
2276 \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`.

```

2277 \newskip\eledsection@correcting@skip

```

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

```

2278 \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 toolbox list, which contains the lines in which page breaks occur (before or after). The `\l@prev@nopbR` macro is a toolbox list, which contains the lines in which NO page breaks occur (before or after).

```

2279 \def\l@prev@pbR{}
2280 \def\l@prev@nopbR{}

```

`\ledpbR` The `\ledpbR` macro writes the call to `\led@pbR` in line-list file. The `\ledpbnR` macro writes the call to `\led@pbnR` in line-list file. The `\lednopbR` macro writes the call to `\led@nopbR` in line-list file. The `\lednopbnR` macro writes the call to `\led@nopbnR` in line-list file.

```
2281 \newcommand{\ledpbR}{\write\linenum@outR{\string\led@pbR}}
2282 \newcommand{\ledpbnR}[1]{\write\linenum@outR{\string\led@pbnR{#1}}}
2283 \newcommand{\lednopbR}{\write\linenum@outR{\string\led@nopbR}}
2284 \newcommand{\lednopbnR}[1]{\write\linenum@outR{\string\led@nopbnR{#1}}}
```

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

```
2285 \newcommand{\led@pbR}{\listcsxadd{1@prev@pbR}{\the\absline@numR}}
2286 \newcommand{\led@pbnR}[1]{\listcsxadd{1@prev@pbR}{#1}}
2287 \newcommand{\led@nopbR}{\listcsxadd{1@prev@nopbR}{\the\absline@numR}}
2288 \newcommand{\led@nopbnR}[1]{\listcsxadd{1@prev@nopbR}{#1}}
```

30 Parallel ledgroup

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

```
2289 \newmarks\parledgroup@
2290 \newmarks\parledgroup@series
2291 \newmarks\parledgroup@type
```

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

```
2292 \newcommand{\parledgroup@notes@startL}{%
2293   \ifnum\usenamecount{1@dmaxlinesinpar\the\l@dpscL}>0%
2294     \IfStrEq{\splitfirstmarks\parledgroup@type}{footnoteX}{\csuse{hooknoteX@\splitfirstmarks\parledgroup@type}{footnoteX}}{\csuse{hooknoteX@\splitfirstmarks\parledgroup@type}{Xfootnote}}
2295     \IfStrEq{\splitfirstmarks\parledgroup@type}{Xfootnote}{\csuse{hookXnote@\splitfirstmarks\parledgroup@type}{Xfootnote}}{\csuse{hooknoteX@\splitfirstmarks\parledgroup@type}{footnoteX}}
2296   \fi%
2297   \global\ledgroupnotesL@true%
2298   \insert@noterule@ledgroup{L}%
2299 }
2300 \newcommand{\parledgroup@notes@startR}{%
2301   \ifnum\usenamecount{1@dmaxlinesinpar\the\l@dpscR}>0%
2302     \IfStrEq{\splitfirstmarks\parledgroup@type}{footnoteX}{\csuse{hooknoteX@\splitfirstmarks\parledgroup@type}{footnoteX}}{\csuse{hooknoteX@\splitfirstmarks\parledgroup@type}{Xfootnote}}
2303     \IfStrEq{\splitfirstmarks\parledgroup@type}{Xfootnote}{\csuse{hookXnote@\splitfirstmarks\parledgroup@type}{Xfootnote}}{\csuse{hooknoteX@\splitfirstmarks\parledgroup@type}{footnoteX}}
2304   \fi%
2305   \global\ledgroupnotesR@true%
2306   \insert@noterule@ledgroup{R}%
2307 }
```

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

```
2308 \newcommand{\parledgroup@notes@endL}{%
2309   \global\ledgroupnotesL@false%
2310 }
2311 \newcommand{\parledgroup@notes@endR}{%
2312   \global\ledgroupnotesR@false%
2313 }
```

`\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`

```
2314 \newcommand{\insert@noterule@ledgroup}[1]{
2315   \IfStrEq{\splitbotmarks\parledgroup@}{begin}{%
2316     \IfStrEq{\splitbotmarks\parledgroup@type}{Xfootnote}{
2317       \csuse{ifledgroupnotes#1@}
2318       \vskip\skip\csuse{mp\splitbotmarks\parledgroup@series footins}
2319       \csuse{\splitbotmarks\parledgroup@series footnoterule}
2320       \fi
2321     }
2322   {}
2323   \IfStrEq{\splitbotmarks\parledgroup@type}{footnoteX}{
2324     \csuse{ifledgroupnotes#1@}
2325     \vskip\skip\csuse{mpfootins\splitbotmarks\parledgroup@series}
2326     \csuse{footnoterule\splitbotmarks\parledgroup@series}
2327     \fi
2328   }{}
2329 }
2330 {}
2331 }
```

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

```
2332 \newcommand{\parledgroupnotespadding}{}%
```

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

```
2333 \dimdef{\parledgroup@notespadding@correction}{0pt}
2334 \newcommand{\parledgroup@notespadding@set@correction}{%
2335   {\notefontsetup\parledgroupnotespadding\dimgdef{\temp@spacing}{\baselineskip}}%
2336   \dimgdef{\parledgroup@notespadding@correction}{\baselineskip-\temp@spacing}%
2337 }
```

`\parledgroup@correction@notespadding@init` `\parledgroup@correction@notespadding@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.

```
2338 \newcommand{\parledgroup@correction@notespadding@init}{%
```



```

2339 \dimdef{\parledgroup@notespacing@correction@accumulated}{0pt}
2340 \dimdef{\parledgroup@notespacing@correction@modulo}{0pt}
2341 }
2342 \parledgroup@correction@notespacing@init

```

`\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`.

```

2343 \newcommand{\parledgroup@correction@notespacing@final}[1]{
2344   \ifparledgroup
2345   \vspace{\parledgroup@notespacing@correction@accumulated}
2346   \parledgroup@correction@notespacing@init%
2347   \ifstrequal{#1}{L}{
2348     \numdef{\@checking}{\the\l@dpscL-1}
2349   }{
2350     \numdef{\@checking}{\the\l@dpscR-1}
2351   }
2352   \dimdef{\@beforenotes@current@diff}{\csuse{@parledgroup@beforenotes@\@checking L}-\csuse{@parledgroup@beforenotes@\@checking R}}
2353   \ifstrequal{#1}{L}{
2354     {% Left
2355       \ifdimgreater{\@beforenotes@current@diff}{0pt}{\vspace{-\@beforenotes@current@diff}}%
2356     }%
2357     {% Right
2358       \ifdimgreater{\@beforenotes@current@diff}{0pt}{\vspace{\@beforenotes@current@diff}}%
2359     }%
2360   \fi
2361 }

```

`\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.

```

2362 {}
2363 \newcommand{\parledgroup@correction@notespacing}[1]{%
2364   \csuse{ifledgroupnotes#1@}%
2365   \vspace{-\parledgroup@notespacing@correction}%
2366   \dimdef{\parledgroup@notespacing@correction@accumulated}{\parledgroup@notespacing@correction@accumulated}
2367   \dimdef{\parledgroup@notespacing@correction@modulo}{\parledgroup@notespacing@correction@modulo}

```

```

2368     \ifdimless{\parledgroup@notespacing@correction@modulo}{\baselineskip}{\advance\num
2369     \dimdef{\parledgroup@notespacing@correction@modulo}{\parledgroup@notespacing@correct
2370     }% mean greater than equal
2371     \fi%
2372 }

```

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

```

2373 \dimdef\parledgroup@beforenotesL{0pt}
2374 \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.

```

2375 \newcommand{\parledgroup@beforenotes@save}[1]{
2376   \ifparledgroup
2377     \csdimgdef{@parledgroup@beforenotes@the\csuse{1@dnumstarts#1}#1}{\csuse{parledgroup
2378     \csdimgdef{parledgroup@beforenotes#1}{0pt}
2379   \fi
2380 }

```

31 The End

`i/codei`

Appendix A Some things to do when changing version

Appendix A.1 Migration to eledpar 1.4.3

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

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

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

See the two following examples:

With standard `\hangingsymbol`:

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

With the modification of `\hangingsymbol`:

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

References

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

Index

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

Symbols	<code>\@adv</code>	361, 626, 627
<code>\&</code>	1556, 1557, 1561, 1578, 1599	<code>\@afterindentfalse</code> 751
<code>\@M</code>	1567	<code>\@arabic</code> 217, 218, 800, 803

- `\advanceline` 625, 656
`\affixline@num` 967
`\affixline@numR` 1032, 1158
`\affixpstart@numL` 981, 1265
`\affixpstart@numR` 1265
`\affixside@note` 983, 988
`\affixside@noteR` 1437
`\aftercolumnseparator` . 4, 1804, 1845
`\appto` 1443, 1444
`\araw@textfalse` 1862
`\araw@texttrue` 1862
`astanza (environment)` 8, 1559
`\AtBeginDocument` ... 1414, 1649, 1675
- B**
- `\ballast@count` 1086, 1091
`\bbl@main@language` 1691, 1692
`\bbl@set@language` 1682, 1683
`\beforecolumnseparator`
 4, 1798, 1801, 1807, 1845
`\beginnumbering` 7, 775, 814
`\beginnumberingR` ... 42, 118, 775, 858
`\bfseries` 800, 803
`\bypage@Rfalse` 138, 153, 158
`\bypage@Rtrue` 138, 148
`\bypstart@Rfalse` 138, 149, 159
`\bypstart@Rtrue` 138, 154
- C**
- `\c@ballast` 1091
`\c@chapter` 100
`\c@chapterR` 100
`\c@firstlinenumR` 182, 1231
`\c@firstsublinenumR` 186, 1226
`\c@linenumincrementR` 182, 1231
`\c@page` 609, 611, 2025, 2031, 2034, 2041
`\c@pstart` 1403
`\c@pstartL` 800
`\c@pstartR` 803, 1391
`\c@section` 101
`\c@sectionR` 101
`\c@sublinenumincrementR` .. 186, 1226
`\c@subsection` 102
`\c@subsectionR` 102
`\c@subsubsection` 103
`\c@subsubsectionR` 103
`\ch@ck@l@ckR` 1158
`\ch@cksub@l@ckR` 1158
`\ch@cksub@lockR` 1227
`\chapter` 737, 738, 747
- `\chapterinpages` 729, 738, 749
`\chardef` 1556
`\check@goal` 2100, 2114, 2171
`\check@pstarts`
 1725, 1774, 1850, 1911, 1919, 1927
`\checkpageL` 1938, 1957, 2093
`\checkpageR` 1969, 1988, 2093
`\checkpb@columns` ... 1770, 1812, 1816
`\checkpbL` 1959, 2126
`\checkpbR` 1990, 2126
`\checkraw@text`
 1731, 1767, 1862, 1935, 2000
`\checkverseL` 1768, 1958, 2149
`\checkverseR` 1769, 1989, 2149
`\cleardoublepage` 2037
`\clearl@dleftpage` 1968, 2023
`\clearl@drighpage` 1999, 2023
`\cleartoevenpage` 2023
`\cleartol@devenpage` 1900, 2023
`\closeout` 92, 590, 596, 599, 603
`\columnrulewidth` 4, 1836
`\Columns` 3, 1710
`\columns@position` .. 1755, 1765, 1841
`\columnseparator` 4, 1803, 1836
`\columnsposition` 4, 1841
`\correcthangingL` 985, 1537
`\correcthangingR` 1537
`\countLline` 938, 949
`\countRline` 938, 1014
`\critext` 661
`\cs` 1813, 2206, 2222
`\csdimgdef` 2377, 2378
`\csexpandonce` .. 1487, 1491, 1507, 1518
`\csgdef` 847, 890, 911, 932
`\csundef` 997, 1740, 1751,
 1944, 1975, 2209, 2226, 2255, 2267
`\csuse` 995,
 1506, 1517, 1738, 1749, 1943,
 1974, 2208, 2225, 2254, 2266,
 2294, 2295, 2302, 2303, 2317–
 2319, 2324–2326, 2352, 2364, 2377
- D**
- `\DeclareOption` 8–10
`\def@tempb` 156
`\dimdef` 2333, 2339, 2340,
 2352, 2366, 2367, 2369, 2373, 2374
`\dimen` 612, 613, 617–619, 623
`\dimgdef` 2335, 2336
`\divide` 1163

- \do@actions 1068
 - \do@actions@fixedcodeR 1095
 - \do@actions@nextR 1095
 - \do@actionsR 1050, 1095
 - \do@ballast 1069
 - \do@ballastR 1051, 1086
 - \do@insidelineLhook 985, 1007
 - \do@insidelineRhook 1007
 - \do@lineL 948, 1734, 1945
 - \do@lineLhook 953, 1007
 - \do@lineR 1012, 1745, 1976
 - \do@lineRhook 1007, 1018
 - \do@lockoff 503
 - \do@lockoffL 527
 - \do@lockoffR 503
 - \do@lockon 468
 - \do@lockonL 500
 - \do@lockonR 468
 - \dolistloop 1448, 1468, 1475
 - \dummy@ref 549
- E**
- \edfont@info 712, 715, 721, 724
 - \edlabel 1379
 - \edtext 684
 - \eled@sectioningR@out .. 69, 92, 2278
 - \eled@sections@@ 968, 993, 1735, 1946
 - \eled@sectionsR@@ 67, 1033, 1746, 1977
 - \eledpar@error 21, 30, 33, 35
 - \eledsection@correcting@skip ..
..... 1005, 1711, 1891, 2277
 - \eledsectmark 13, 2275
 - \eledsectnotoc 13, 2273
 - \empty .. 78, 81, 267, 275, 674, 697,
710, 719, 823, 867, 1148, 1230,
1238, 1334–1336, 1347, 1358,
1382, 1394, 2048, 2054, 2061, 2067
 - \end@lemmas 674, 675, 697, 698
 - \endashchar 1369
 - \endgraf 900, 921, 1719, 1903
 - \endline@num 556, 562
 - \endlock 645, 1565, 1574, 1579
 - \endnumbering 7, 71, 122, 776
 - \endnumberingR 45, 71, 106, 117, 130, 776
 - \endpage@num 555, 562
 - \endstanzaextra 1581
 - \endsub 612
 - \endsubline@num 557, 563
 - \enlargethispage 1833
- environments:
- astanza 8, 1559
 - Leftside 5, 756
 - pages 4, 729
 - pairs 3, 729
 - Rightside 5, 773
 - \extensionchars 62, 112, 127, 135
- F**
- \f@x@l@cksR 1158
 - \first@linenum@out@Rfalse .. 585, 591
 - \first@linenum@out@Rtrue 585
 - \firstlinenum 5, 191
 - \firstlinenum* 5, 191
 - \firstsublinenum 5, 191
 - \firstsublinenum* 5, 191
 - \fix@page 328, 335
 - \flag@end
..... 612, 670, 680, 681, 693, 703, 704
 - \flag@start
..... 612, 669, 670, 681, 692, 693, 704
 - \flush@notes 1786, 2009
 - \flush@notesR 1356, 1787, 2010
 - \footnotelang@lua 1501, 1512
 - \footnotelang@poly 1505, 1516
 - \fullstop . 230, 1366, 1368, 1370, 1372
- G**
- \get@linelistfile 263
 - \get@nextboxL 1956, 2184
 - \get@nextboxR 1987, 2184
 - \getline@numL 959, 1066
 - \getline@numR 1024, 1048
 - \getlinesfrompagelistL
..... 1923, 2002, 2060
 - \getlinesfrompagelistR
..... 1924, 2003, 2060
 - \getlinesfromparlistL ... 1915, 2047
 - \getlinesfromparlistR ... 1916, 2047
 - \gl@p 270, 271,
278, 279, 675, 698, 714, 723,
1151, 1152, 1340, 1344, 1359,
1385, 1397, 2051, 2057, 2064, 2070
 - \goalfraction 5, 2171
- H**
- \hangingsymbol 10, 1528, 1534
 - \hb@xt@ 966, 975, 985, 1031,
1041, 1754, 1951, 1954, 1982, 1985

\hsize 841,
884, 1754, 1951, 1954, 1982, 1985

I

\if@filesw 1663
\if@firstcolumn 1244, 1269, 1290, 1454
\if@ledgroup 595
\if@nobreak 806, 850
\if@nopb 1818, 1833
\if@pb 1817, 1834
\if@pstarts 1726, 1850, 1912, 1928
\if@RTL 670, 681, 693, 704, 998
\ifaraw@text ... 1732, 1862, 1937, 2001
\ifautopar 833, 877
\ifbypage@ 351
\ifbypage@R 138, 341, 1130
\ifbypstart@ 570, 1775, 2199
\ifbypstart@R ... 138, 574, 1779, 2247
\ifdefstring 990, 991,
1737, 1748, 1755, 1765, 1942, 1973
\ifdim ... 613, 617, 619, 623, 1798,
1804, 1951, 1982, 2027, 2101, 2115
\ifdimgreater 2355, 2358
\ifdimless 2368
\iffirst@linenum@out@R 585, 589
\ifhbox 1757, 1762
\ifinserthangingsymbol
..... 1526, 1540, 2152
\ifinserthangingsymbolR
..... 1524, 1532, 1550, 2163
\ifinstanzaL 754, 754, 1527, 1539, 2150
\ifinstanzaR 754, 755, 1533, 1549, 2161
\ifl@d@dash 1369
\ifl@d@elin 1371, 1372
\ifl@d@esl 1372
\ifl@d@pnum 1366, 1370
\ifl@d@ssub 1368
\ifl@dpagefull 1961, 1992, 2093
\ifl@dpadding 12, 1538, 1548
\ifl@dpairing 12, 75
\ifl@dsamelang 1660
\ifl@dsamepage 1941, 1971, 2093
\ifl@dskipnumber 1221
\ifl@dusedbabel 1658
\iflabelpstart 843, 886
\ifledgroupnotesL@ 1070
\ifledgroupnotesR@ 1052, 1220
\iflednopbinverse 2151, 2162
\ifledplinenum 1367

\ifledRcol 12, 174,
196, 200, 204, 208, 250, 265,
329, 338, 363, 377, 394, 411,
423, 431, 452, 497, 524, 533,
542, 614, 620, 626, 633, 641,
646, 650, 655, 665, 688, 709,
1380, 1419, 1486, 1499, 1686, 1698
\ifluatex 1500, 1511
\ifnoteschanged@ 85
\ifnumberedpar@
... 816, 860, 896, 917, 1484, 1498
\ifnumbering 143, 812, 893
\ifnumberingR ... 43, 72, 108, 856, 914
\ifnumberline 1052, 1070, 1220
\ifnumberpstart ... 834, 878, 905, 926
\ifnumequal 1441
\ifnumgreater 1449, 1469, 1476
\ifodd 1254, 1279,
1300, 1464, 2025, 2031, 2034, 2041
\ifparledgroup 2344, 2376
\ifpst@rtedL 38, 820
\ifpst@rtedR 38, 864
\ifpstartnum 1310, 1315
\ifpstartnumR 1265
\ifshiftedpstarts 5, 1950, 1981, 2172
\ifsidepstartnum 835, 879, 1267, 1288
\ifstreempty 845, 888, 909, 930
\IfStrEq 957, 1022, 1819,
1827, 2127, 2131, 2139, 2143,
2154, 2155, 2165, 2166, 2294,
2295, 2302, 2303, 2315, 2316, 2323
\ifstrequal 2347, 2353
\ifsublines@ 228,
317, 362, 395, 402, 433, 442,
454, 461, 473, 507, 561, 563,
1053, 1071, 1137, 1224, 1421, 1425
\ifvbox 950, 1015, 1866, 1869, 2185, 2233
\ifvmode 1387, 1399
\ifwrittenlinesL 2181, 2193
\ifwrittenlinesR 2182, 2241
\initnumbering@sectcmd 114, 732, 741
\initnumbering@sectcountR 66, 95, 1972
\InputIfFileExists 68
\insert@count 539, 666, 689, 1493, 1520
\insert@countR 540, 665, 688, 1489, 1509
\insert@noterule@ledgroup
..... 2298, 2306, 2314
\inserthangingsymbolfalse 963
\inserthangingsymbolL 985, 1524
\inserthangingsymbolR 1524

- \inserthangingsymbolRfalse 1028
 - \inserthangingsymbolRtrue 1026
 - \inserthangingsymboltrue 961
 - \insertlines@listR
 - 78, 238, 252, 545, 1336, 1340
 - \inserts@list 822, 1492, 1519
 - \inserts@listR 866, 1331,
 - 1334, 1344, 1358, 1359, 1488, 1508
 - \istanzaLfalse 1794, 2016
 - \istanzaLtrue 765
 - \istanzaRfalse 1795, 2017
 - \istanzaRtrue 788
 - \interlinepenalty 1567
 - \itemcount@ 1439, 1441,
 - 1446, 1449, 1467, 1469, 1474, 1476
- L**
- \l@d@nums 712, 715, 721, 724
 - \l@d@set 410, 641, 642
 - \l@dbbl@set@language 1660, 1683
 - \l@dbfnote 1483
 - \l@dc@maxchunks 828, 830,
 - 872, 874, 1617, 1627, 1635, 1642
 - \l@dcalc@maxoftwo
 - 1917, 2080, 2214, 2258
 - \l@dcalc@minoftwo .. 1925, 2004, 2080
 - \l@dcalcnun 1158
 - \l@dchecklang 1660
 - \l@dchset@num 284, 287, 410
 - \l@dcsnotetext 1448, 1451
 - \l@dcsnotetext@l 1451, 1468
 - \l@dcsnotetext@r 1451, 1475
 - \l@demptyd@ta 954, 1019
 - \l@dend@stuff 63, 113, 128, 136
 - \l@dgetline@margin 172
 - \l@dgetsidenote@margin 1430
 - \l@dld@ta 982,
 - 1245, 1257, 1270, 1282, 1291, 1303
 - \l@dleftbox
 - .. 935, 965, 975, 1756, 1951, 1954
 - \l@dlinenumR 220
 - \l@dlsn@te 984
 - \l@dmake@labels 1403
 - \l@dmake@labelsR 1391, 1408
 - \l@dminpagelines
 - 1886, 1926, 2005, 2102, 2116
 - \l@dnumpstartsL
 - 827, 828, 830, 832, 847,
 - 911, 1621, 1653, 1714, 1715,
 - 1791, 1853, 1897, 1898, 2014, 2197
 - \l@dnumpstartsR
 - .. 47, 871, 872, 874, 876, 890,
 - 932, 1621, 1654, 1714, 1715,
 - 1792, 1856, 1897, 1898, 2015, 2245
 - \l@doldbbl@set@language 1682
 - \l@doldselectlanguage 1681, 1685, 1690
 - \l@dpagfullfalse
 - 2093, 2129, 2134, 2141, 2146
 - \l@dpagfulltrue 2093,
 - 2128, 2133, 2140, 2145, 2224, 2268
 - \l@dpagingfalse 14, 731, 746
 - \l@dpagingtrue 740
 - \l@dpairingfalse 12, 734, 745
 - \l@dpairingtrue 730, 739
 - \l@dpscl 950,
 - 955, 968, 992, 995, 997, 1623,
 - 1655, 1723, 1729, 1735, 1738,
 - 1740, 1789, 1853, 1866, 1907,
 - 1913, 1918, 1921, 1929, 1943,
 - 1944, 1946, 2012, 2185, 2187,
 - 2190, 2197, 2208, 2209, 2214,
 - 2216, 2218, 2225, 2226, 2293, 2348
 - \l@dpscR 1015, 1020, 1033,
 - 1624, 1656, 1724, 1730, 1746,
 - 1749, 1751, 1790, 1856, 1869,
 - 1908, 1914, 1922, 1930, 1974,
 - 1975, 1977, 2013, 2233, 2235,
 - 2238, 2245, 2254, 2255, 2258,
 - 2260, 2262, 2266, 2267, 2301, 2350
 - \l@drrd@ta 985,
 - 1247, 1255, 1272, 1280, 1293, 1301
 - \l@drightrightbox
 - 935, 1030, 1041, 1761, 1982, 1985
 - \l@drrsn@te 986
 - \l@dsamepagefalse
 - 2093, 2128, 2133, 2140, 2145
 - \l@dsamepagetrue
 - 2093, 2129, 2134, 2141, 2146
 - \l@dsetupmaxlinecounts .. 1634, 1651
 - \l@dsetuprawboxes 1626, 1650
 - \l@dskipnumberfalse 1222
 - \l@dskipnumbertrue 1118
 - \l@dunhbox@line 985, 1000, 1003
 - \l@dusedbabelfalse 1658, 1678
 - \l@dusedbabeltrue 1658, 1680
 - \l@duselanguage
 - 1670, 1733, 1744, 1939, 1970
 - \l@dzeromaxlinecounts ... 1634, 1652
 - \l@dzeropenalties 899, 920, 1718, 1902
 - \l@prev@nopbR 54, 2280

- \l@prev@pbR 53, 2279
- \l@pscL 1623
- \l@pscR 1623
- \label@refs
 - 1383, 1385, 1391, 1395, 1397, 1403
- \labelref@list 1394, 1397, 1426
- \labelref@listR 1377, 1382, 1385, 1422
- \language name .. 1661, 1662, 1664–1667
- \last@page@num 349, 355
- \last@page@numR 335
- \lastbox 958, 1023
- \lastskip 612, 618
- \lcolwidth
 - . 4, 5, 16, 742, 841, 966, 975, 1741
- \led@err@BadLeftRightPstarts ...
 - 29, 1715, 1898
- \led@err@LeftOnRightPage ... 32, 2035
- \led@err@LineationInNumbered ... 144
- \led@err@ManyLeftnotes 1469
- \led@err@ManyRightnotes 1476
- \led@err@ManySidenotes 1449
- \led@err@NumberingNotStarted ... 89
- \led@err@numberingShouldHaveStarted
 - 116
- \led@err@NumberingStarted 44
- \led@err@PendNoPstart 897, 918
- \led@err@PendNotNumbered ... 894, 915
- \led@err@PstartInPstart ... 817, 861
- \led@err@PstartNotNumbered . 813, 857
- \led@err@RightOnLeftPage ... 32, 2042
- \led@err@TooManyPstarts
 - 24, 26, 829, 873
- \led@mess@NotesChanged 86
- \led@mess@SectionContinued
 - 111, 126, 134
- \led@nopbnumR 2284, 2285
- \led@nopbR 2283, 2285
- \led@pb@setting
 - 1819, 1827, 2127, 2131,
 - 2139, 2143, 2154, 2155, 2165, 2166
- \led@pbnumR 2282, 2285
- \led@pbR 2281, 2285
- \led@warn@BadAction 1120
- \led@warn@BadAdvancelineLine 380, 386
- \led@warn@BadAdvancelineSubline .
 - 366, 372
- \led@warn@BadLineation 161
- \led@warn@BadSetline 631
- \led@warn@BadSetlinenum 639
- \led@warn@DuplicateLabel 1410
- \ledgroupnotesL@false 2309
- \ledgroupnotesL@true 2297
- \ledgroupnotesR@false 2312
- \ledgroupnotesR@true 2305
- \ledllfill 985
- \lednopb 784
- \lednopbnum 2154, 2281
- \lednopbnumR 2165, 2281
- \lednopbR 784, 2283
- \ledpb 783
- \ledpbnum 2155
- \ledpbnumR 2166, 2281
- \ledpbR 783, 2281
- \ledRcol@false 1044
- \ledRcol@true 1013
- \ledRcolfalse 15, 757, 790
- \ledRcoltrue 774
- \ledrlfill 985
- \ledsavedprintlines 7, 1364
- \ledsectnomark 991
- \ledsectnotoc 990, 1942, 1973
- \ledstrutL 1951, 1954, 2020
- \ledstrutR 1982, 1985, 2020
- \ledthegoal 2101, 2115, 2171
- \leftlinenumR 220, 1245, 1257
- \leftpstartnumL 1265
- \leftpstartnumR 1265
- Leftside (environment) 5, 756
- \Leftsidehook 763, 768
- \Leftsidehookend 767, 768
- \line@list 719, 723
- \line@list@stuff 127
- \line@list@stuffR .. 62, 112, 135, 587
- \line@listR . 81, 238, 251, 563, 710, 714
- \line@margin 177, 1275
- \line@marginR 170, 1250, 1296
- \line@num . 352, 384, 385, 387, 405,
 - 416, 417, 445, 570, 1077, 1424, 2202
- \line@numR 55,
 - 227, 234, 289, 323, 342, 378,
 - 379, 381, 398, 412, 413, 436,
 - 556, 560, 574, 1059, 1131, 1140,
 - 1229, 1231, 1233, 1234, 1420, 2250
- \lineation 166, 167, 785
- \lineation* 6, 166
- \lineationR 6, 142, 168, 785
- \linenum@out
 - . 609, 615, 621, 627, 634, 642,
 - 647, 651, 1393, 1776, 1877, 2074

- \linenum@outR
 - . 584, 590, 592, 596, 597, 599,
 - 600, 603, 604, 611, 614, 620,
 - 626, 633, 641, 646, 650, 655,
 - 1381, 1780, 1882, 2077, 2281–2284
 - \linenumberlist 1230, 1234
 - \linenumincrement 5, [191](#)
 - \linenumincrement* 5, [191](#)
 - \linenummargin [170](#)
 - \linenumr@p 1367, 1371, 1420, 1424
 - \linenumrepR [217](#), 227
 - \linenumsep
 - . 222, 224, 1312, 1315, 1324, 1327
 - \linesinpar@listL
 - [243](#), 259, 571, 2048, 2051
 - \linesinpar@listR
 - [243](#), 255, 575, 2054, 2057
 - \linesonpage@listL 260, 579, 2061, 2064
 - \linesonpage@listR 256, 582, 2067, 2070
 - \list@clear
 - . 251–256, 259, 260, 262, 822, 866
 - \list@clearing@reg 258
 - \list@create
 - ... 238–241, 243–245, 1331, 1377
 - \listcsxadd 357, 2285–2288
 - \lock@disp 1190, 1194, 1199
 - \lock@off 494, 495, [503](#), 650, 651
 - \lock@on 646, 647
- M**
- \managestanza@modulo 1593
 - \maxchunks 3, [1617](#)
 - \maxlinesinpar@list [243](#), 262
 - \memorydump 7, 762, 779
 - \memorydumpL [121](#), 762
 - \memorydumpR [121](#), 779
 - \message 61
 - \multiply 1164
- N**
- \n@num [531](#), 655
 - \n@num@reg 537
 - \namebox 950, 955, 1015,
 - 1020, [1601](#), 1866, 1869, 2185, 2233
 - \NeedsTeXFormat 2
 - \new@line 1000, 1003
 - \new@lineL [608](#), 985
 - \new@lineR [610](#)
 - \newbox 795, 935, 936, 1602
 - \newcommandx 805, 849, 892, 913
 - \newcounter 95–98, 182,
 - 184, 186, 188, 798, 799, 801, 802
 - \newif 5,
 - 13, 39, 138, 139, 585, 754, 755,
 - 1319, 1524, 1658, 1817, 1818,
 - 1850, 1862, 2093, 2095, 2181, 2182
 - \newlength 1845, 1848
 - \newmarks 2289–2291
 - \newnamebox [1601](#), 1629, 1630
 - \newnamecount [1612](#), 1637
 - \newsavebox 1708, 1709
 - \newwrite 584, 2278
 - \next@absline
 - 1820, 1822, 1824, 2132–2134
 - \next@abslineR
 - 1821, 1823, 1825, 2144–2146
 - \next@action 279
 - \next@actionline 276, 278
 - \next@actionlineR
 - . 268, 270, 1089, 1127, 1149, 1151
 - \next@actionR 271, 1090,
 - 1128, 1129, 1134, 1135, 1143, 1152
 - \next@insert 823
 - \next@insertR
 - 867, 1335, 1338, 1340, 1343, 1347
 - \next@page@num 356, 428
 - \next@page@numR 59, 292, 294, 346, 425
 - \no@expands 663, 686
 - \noindent 847, 890, 911, 932
 - \normal@page@breakR 52
 - \normal@pars 74, 826, 870
 - \normalbfnoteX [1497](#)
 - \notefontsetup 2335
 - \noteschanged@true
 - 79, 82, 711, 720, 1337
 - \num@lines 900, 1719, 1903
 - \num@linesR [794](#), 921, 1720, 1904
 - \numberedpar@true 842, 885
 - \numberingRfalse 73
 - \numberingRtrue 49, 106, 131
 - \numberingtrue 123
 - \numberpstartfalse 8
 - \numberpstarttrue 8
 - \numdef 992, 1467, 1474,
 - 1820, 1821, 2132, 2144, 2348, 2350
 - \numgdef 1439, 1446, 2153, 2164
 - \numlabfont 227
 - \numpagelinesL [1886](#),
 - 1949, 1962, 1966, 2102, 2155, 2368

- `\numpagelinesR`
 1886, 1980, 1993, 1997, 2116, 2166
- O**
- `\old@otherlanguage` 1695
`\old@startstanza` .. 764, 765, 787, 788
`\oldchapter` 737, 747
`\one@line` ... 955, 958, 985, 1000, 1003
`\one@lineR` 794, 1020, 1023
`\openout` 69, 592, 597, 600, 604
`\otherlanguage` 1695, 1696
- P**
- `\p@pstartL` 844
`\p@pstartR` 887
`\PackageError` 21
`\page@action` 293, 422, 550
`\page@num` 274, 354, 1277
`\page@numR` 247, 266, 344,
 555, 560, 1129, 1252, 1298, 1462
`\pagebreak` 1834
`\pagegoal` 2179
`\Pages` 4, 1890
`pages` (environment) 4, 729
`\pagetotal` 2027, 2101, 2115
`pairs` (environment) 3, 729
`\paperwidth` 999, 1002
`\par@line` 901, 1721, 1905
`\par@lineR` 794, 922, 1722, 1906
`\parbox` 1741, 1752
`\parledgroup@` .. 957, 1022, 2289, 2315
`\parledgroup@beforenotes@save` ..
 908, 929, 2375
`\parledgroup@beforenotesL` 2373
`\parledgroup@beforenotesR` 2373
`\parledgroup@correction@notespacing`
 1953, 1984, 2362
`\parledgroup@correction@notespacing@final`
 2220, 2264, 2343
`\parledgroup@correction@notespacing@init`
 1967, 1998, 2338, 2346
`\parledgroup@notes@endL`
 2188, 2191, 2219, 2308
`\parledgroup@notes@endR`
 2236, 2239, 2263, 2311
`\parledgroup@notes@startL`
 957, 2292, 2308
`\parledgroup@notes@startR`
 1022, 2292, 2308
`\parledgroup@notespacing@correction`
 2333, 2365–2367
`\parledgroup@notespacing@correction@accumulated`
 2339, 2345, 2366
`\parledgroup@notespacing@correction@modulo`
 2340, 2367–2369
`\parledgroup@notespacing@set@correction`
 1894, 2333
`\parledgroup@series`
 2290, 2294, 2295,
 2302, 2303, 2318, 2319, 2325, 2326
`\parledgroup@type` 2291,
 2294, 2295, 2302, 2303, 2316, 2323
`\parledgroupnotespacing` .. 2332, 2335
`\parledgroupseries@` 2289
`\parledgrouptrue` 10
`\parledgrouptype@` 2289
`\pausenumbering` 777
`\pausenumberingR` 105, 777
`\pend` 5, 761, 782, 818, 1580
`\pendL` 761, 892
`\pendR` 782, 862, 913
`\prev@abslineverse`
 2153–2155, 2164–2166
`\prev@nopbR` 2279
`\prev@pbR` 2279
`\prevgraf`
 900, 921, 1719, 1720, 1903, 1904
`\print@columnseparator` .. 1760, 1797
`\print@eledsectionL`
 980, 987, 1741, 1947
`\print@eledsectionR` .. 1048, 1752, 1978
`\print@lineL` 970, 980
`\print@lineR` 1035, 1048
`\printlines` 1375
`\printlinesR` 7, 1364
`\ProcessOptions` 11
`\protected@csxdef` 1506, 1517
`\protected@edef` 843, 886
`\protected@write` ... 1390, 1402, 1664
`\ProvidesPackage` 3
`\pst@rtedLfalse` 38
`\pst@rtedLtrue` 124, 824
`\pst@rtedRfalse` 40, 48, 76
`\pst@rtedRtrue` 109, 132, 868
`\pstart` 5, 27, 31, 759, 781, 1582
`\pstartL` 759, 797
`\pstartnumfalse` 1312, 1317
`\pstartnumRfalse` 1324, 1329
`\pstartnumRtrue` 1320, 1728, 2257

- `\pstartnumtrue` 1727, 2213
`\pstartR` 781, 797
- R**
- `\Rcolwidth` 4,
 5, 16, 743, 884, 1031, 1041, 1752
`\read@linelist` 249, 588
`\rem@inder` 1234, 1236–1238
`\resetprevline@` 1777, 1781, 2203, 2251
`\resumenumbering` 778
`\resumenumberingR` 105, 778
`\rightlinenumR` 220, 1247, 1255
`\rightpstartnumL` 1265
`\rightpstartnumR` 1265
`Rightside` (environment) 5, 773
`\Rightsidehook` 768, 786
`\Rightsidehookend` 768, 791
`\rlap` 1247, 1255, 1272, 1280, 1293, 1301
`\Rlineflag` 7, 215, 227, 1367, 1371, 1412
`\rule` 1837
- S**
- `\savebox` 1741, 1752
`\sc@n@list` 1235, 1237
`\secdef` 752
`\section@num` 125–127
`\section@numR` 36,
 50, 61, 62, 68, 69, 110–112, 133–135
`\select@language` ... 1662, 1664–1667
`\selectlanguage` 1670
`\set@line` 664, 687, 708
`\set@line@action`
 286, 391, 400, 407, 430, 552
`\setl@dlp@rbox` 1455, 1470, 1472
`\setl@drp@rbox` 1457, 1465, 1477
`\setline` 629
`\setlinenum` 637
`\setnamebox` 832, 876, 1601
`\setprintlines` 1365
`\shiftedpstartsfalse` 7
`\shiftedpstartstrue` 6, 8, 9
`\shiftedversesfalse` 7
`\shiftedversestrue` 6
`\showlemma` 673, 696
`\sidenote@margin` 1432, 1435
`\sidenote@marginR` 1429, 1460
`\sidenotecontent@`
 . 1438, 1443, 1444, 1455, 1457,
 1465, 1466, 1470, 1472, 1473, 1477
`\sidenotemargin` 1429
- `\sidenotemargin*` 1429
`\sidenotesep` 1444
`\skip` 2318, 2325
`\skip@lockoff` 495, 503
`\skipnumbering` 8, 654
`\skipnumbering@reg` 658
`\smash` 1837
`\splitbotmarks` 2315,
 2316, 2318, 2319, 2323, 2325, 2326
`\splitfirstmarks`
 957, 1022, 2294, 2295, 2302, 2303
`\splittopskip` 952, 1017
`\stanza@count` 1562, 1576, 1588
`\stanza@chang` 1564, 1596
`\stanza@modulo` 1562, 1591
`\stanzaindentbase`
 1542, 1552, 1589, 1592
`\startlock` 645
`\startstanzahook` 1560
`\startsub` 612
`\sub@action` 302, 451, 551
`\sub@change` 60, 296, 297, 303
`\sub@lock` 1072
`\sub@lockR` 57, 311, 313, 315,
 318, 469, 475, 476, 478, 479,
 509, 510, 512, 1054, 1110, 1112,
 1113, 1115, 1171, 1211, 1213, 1215
`\sub@off` 620, 621
`\sub@on` 614, 615
`\subline@num` 229, 352, 370,
 371, 373, 403, 443, 1073, 1078, 1425
`\subline@numR` 230,
 234, 319, 323, 342, 364, 365,
 367, 396, 434, 557, 561, 1055,
 1060, 1131, 1138, 1225, 1226, 1421
`\sublinenumincrement` 5, 191
`\sublinenumincrement*` 5, 191
`\sublinenumrep` . 1368, 1372, 1421, 1425
`\sublinenumrepR` 217, 230
`\sublines@false` 58, 300, 1100
`\sublines@true` 298, 1098
`\sublock@disp` 1173, 1177, 1182
`\symplinenum` 1367
`\sza@penalty` 1571, 1575
- T**
- `\temp@` 992, 993
`\temp@spacing` 2335, 2336
`\textwidth` 17, 19, 742, 743
`\theledlanguageL` ... 1670, 1733, 1939

<code>\theledlanguageR</code> . . .	1670, 1744, 1970		
<code>\thepage</code>	609, 611, 1391, 1403		
<code>\thepstart</code>	760, 780		
<code>\thepstartL</code>			
. 8, 760, 800, 837, 844, 1311, 1316			
<code>\thepstartR</code>			
. 8, 780, 803, 880, 887, 1323, 1328			
<code>\thr@@</code> . . . 478, 487, 510, 517, 1105, 1113			
<code>\topskip</code>	2027		
U			
<code>\unhbox</code>	1606,		
1756, 1761, 1951, 1954, 1982, 1985			
<code>\unhnamebox</code>	1601		
<code>\unvbox</code>	958, 1023, 1608		
<code>\unvnamebox</code>	1601		
<code>\usebox</code>	1758, 1763		
<code>\usenamecount</code> 1563, 1570, 1612, 1644,			
1918, 2187, 2190, 2214, 2216,			
2235, 2238, 2258, 2260, 2293, 2301			
V			
<code>\value</code>	821, 865, 1587,		
1712, 1713, 1892, 1893, 2200, 2248			
<code>\vbadness</code>	951, 1016		
<code>\vbfnoteX</code>	1507, 1518		
<code>\vbox</code>	832, 876, 1741, 1752		
<code>\vl@dbfnote</code>	1487, 1491		
<code>\vsplit</code>	955, 1020		
W			
<code>\W</code>	2278		
<code>\wd</code>	985		
<code>\WithSuffix</code>	166, 211–214, 1429		
<code>\writtenlinesLfalse</code>	1909, 2198		
<code>\writtenlinesLtrue</code>	2195		
<code>\writtenlinesRfalse</code>	1910, 2246		
<code>\writtenlinesRtrue</code>	2243		
X			
<code>\x@lemma</code>	675–677, 698–700		
<code>\xifinlist</code>	968,		
993, 1033, 1735, 1746, 1946, 1977			
<code>\xifinlistcs</code>	1822–		
1825, 1828–1831, 2128, 2129,			
2133, 2134, 2140, 2141, 2145, 2146			
<code>\xpg@main@language</code>	1702, 1703		
<code>\xpg@set@language</code>	1701		
<code>\xright@appenditem</code>			
.	424, 425, 427, 428,		
432, 439, 441, 448, 453, 455,			
457, 460, 462, 464, 472, 474,			
483, 506, 508, 515, 534, 535,			
545, 559, 571, 575, 579, 582,			
1420, 1424, 1487, 1491, 1507, 1518			
Z			
<code>\zz@@@</code>	1383, 1395		

Change History

v0.1			
General: First public release	1		bering by pstart (like in eledmac 0.15). 41
v0.10			Lineation can be by pstart (like in eledmac 0.15). 17
General: <code>\edlabel</code> commands on the right side are now correctly indicated.	1		New management of hangingsymbol insertion, preventing undesirable insertions. 56
<code>\edlabel</code> commands which start a paragraph are now put in the right place.	1		Prevent shift of column separator when a verse is hanged 56
v0.11			
General: Change <code>\do@lineL</code> and <code>\do@lineR</code> to allow line num-			<code>\affixline@numR</code> : Changed <code>\affixline@numR</code> to allow to disable line numbering (like in

eledmac 0.15).	46	<code>\l@dnumpstartsR</code> :	Moved
<code>\Columns</code> : Line numbering by		<code>\l@dnumpstartsL</code> to eledmac .	59
<code>pstart</code>	64	<code>\ledsavedprintlines</code> :	Simpli-
<code>\get@nextboxR</code> : Change <code>\get@nextboxL</code>		fied <code>\printlinesR</code> by using	
and <code>\get@nextboxR</code> to allow to		<code>\setprintlines</code>	52
disable line numbering (like in		<code>\ledstrutR</code> : Added <code>\ledtrutL</code> and	
eledmac 0.15).	75	<code>\ledstrutR</code>	70
<code>Pstart</code> number can be printed in		<code>\normalbfnoteX</code> :	Removed
side	75	extraneous spaces from	
v0.12		<code>\normalbfnoteX</code>	55
General: New new management of		<code>\Pages</code> : Added <code>\ledstrutL</code> to	
<code>hangingsymbol</code> insertion, pre-		<code>\Pages</code>	68
venting undesirable insertions.	56	Added <code>\ledstrutR</code> to <code>\Pages</code> .	69
v0.2		<code>\Rightsidehookend</code> :	Added
General: Added section of babel re-		<code>\Leftsidehook</code> , <code>\Leftsidehookend</code> ,	
lated code	60	<code>\Rightsidehook</code> and <code>\Rightsidehookend</code>	
Fix babel problems	1	36
<code>\Columns</code> : Added <code>\l@dchecklang</code>		<code>\sublinenumrepR</code> :	Added
and <code>\l@duselanguage</code> to		<code>\linenumrepR</code> and <code>\sublinenumrepR</code>	
<code>\Columns</code>	63	20
<code>\Pages</code> : Added <code>\l@duselanguage</code>		v0.3.a	
to <code>\Pages</code>	68	General: Minor <code>\linenummargin</code>	
v0.3		fix	1
General: Added <code>\do@lineLhook</code>		v0.3.b	
and <code>\do@lineRhook</code>	42	General: Improved parallel page	
Reorganize for ledarab	1	balancing	1
<code>\affixline@numR</code> :	Changed	v0.3.c	
<code>\affixline@numR</code> to match new		General: Compatibilty with Poly-	
eledmac	46	glossia	1
<code>\do@actions@nextR</code> :	Used	v0.3a	
<code>\do@actions@fixedcode</code> in		<code>\line@marginR</code> : Don't just	
<code>\do@actionsR</code>	44	set <code>\line@marginR</code> in	
<code>\do@lineL</code> : Added <code>\do@lineLhook</code>		<code>\linenummargin</code>	19
to <code>\do@lineL</code>	41	v0.3b	
Simplified <code>\do@lineL</code> by using		<code>\Pages</code> : Added <code>\l@dminpagelines</code>	
macros for some common code	41	calculation for succeeding page	
<code>\do@lineR</code> : Changed <code>\do@lineR</code>		pairs	70
similarly to <code>\do@lineL</code>	42	v0.4	
Leftside : Added hooks into Left-		General: No more ledparpatch. All	
side environment	35	patches are now in the main	
<code>\flag@end</code> : Removed extraneous		file.	1
spaces from <code>\flag@end</code>	31	v0.5	
<code>\ifledRcol</code> : Moved <code>\ifl@dpairing</code>		General: Corrections about	
to eledmac	14	<code>\section</code> and other titles in	
<code>\ifpst@rtedR</code> : Moved <code>\ifpst@rtedL</code>		numbered sections	1
to eledmac	15	v0.6	
<code>\l@dlinumR</code> :	Simplified	General: Be able to us <code>\chapter</code> in	
<code>\leftlinenumR</code> and <code>\rightlinenumR</code>		parallel pages.	1
by introducing <code>\l@dlinumR</code>	20		

v0.7	General: Option ‘shiftedverses’ which make there is no blank between two parallel verses with inequal length.	1	v1.1.1	\pstartR: Correct \pstartR bug introduced by 1.1.	37
v0.8	General: Possibility to have a symbol on each hanging of verses, like in the french typography. Redefine the commande \hangingsymbol to define the character.	1	v1.1.2	\affixside@noteR: Remove spurious space between line number and line content	54
v0.9	General: Possibility to number \pstart.	8	v1.12.0	General: \beginnumbering is defined only on eledmac, not on eledpar.	15
	Possilbity to number the pstart with the commands \numberpstarttrue.	1	\Columns: Modify \Columns to enable to add section’s title. . . .	62	
	\ifledRcol: Moved \iflledRcol and \ifnumberingR to eledmac	14	Suppress \l@dchecklang from \Columns.	63	
v0.9.1	General: The numbering of the pstarts restarts on each \beginnumbering.	1	\l@dchecklang: Suppress \l@dchecklang which didn’t work and was not logical, because both columns could have the same language but not the main language of the document.	60	
v0.9.2	General: Debug : with \Columns, the hanging indentation now runs on the left columns and the hanging symbol is shown only when \stanza is used.	1	\Pages: Modify \Pages to enable to add section’s title.	67	
v0.9.3	General: \thepstartL and \thepstartR use now \bfseries and not \bf, which is deprecated and makes conflicts with memoir class.	1	v1.2	General: Support for \led{section} commands in parallel texts. . . .	1
v1.0	General: Compatibility with eledmac. Change name to eledpar. . .	1	v1.2.1	\initnumbering@sectcountR: For the right section, the counter is defined only once.	16
	Debug in lineation by pstart . .	17	v1.3	General: Manage RTL language. . .	33
v1.0.1	General: Correction on \numberonlyfirstinlinenote with lineation by pstart or by page.	1	v1.3.2	General: Debug with some classes. .	1
v1.1	General: Shiftedverses becomes shiftedpstarts.	1	v1.3.3	General: Debugging the left notes of the right column.	54
	\pstartR: Add \labelpstarttrue (from eledmac).	37	\l@dbfnote: Spurious space with footnote in right column. . . .	55	
			v1.3.4	General: Allow use of commands in sidenotes, as introduced by eledmac 1.0.	54
			v1.3.5	\normalbfnoteX: Allows one to redefine \thefootnoteX with alph when some packages are loaded.	55

v1.4	General: Added <code>\do@insidelineLhook</code> and <code>\do@insidelineRhook</code> . . .	42	Add, as in eledmac, new system of sectioning commands.	1
v1.4.1	<code>\normalbfnoteX</code> : Fix bug with normal familiar footnotes when mixing RTL and LTR text. . .	55	Add, as in eledmac, possibility to insert something after <code>\pends</code> / verses.	1
	<code>astanza</code> : Enable the use of stanzaindent repetition within stanza environment.	57	Add, as in eledmac, possibility to insert something between <code>\pstarts</code> / verse.	1
v1.4.2	<code>\line@list@stuffR</code> : Open / close immediatly the line-list file when in minipage, except if the minipage is a ledgroup.	30	Change <code>\do@lineR</code> and <code>\do@lineR</code> to allow new sectioning commands.	41
v1.4.3	General: Corrects a false hanging verse when a verse is exactly the length of a line.	1	Compatibility with musixtex. . .	1
	<code>\inserthangingsymbolR</code> : Hang verse is now not automatically flush right.	56	Debug eledmac sectioning command after using <code>\resumenumbering</code>	1
	<code>\pendL</code> : Spurious spaces in <code>\pendL</code>	39	New sectioning commands, as in eledmac.	13
	<code>\pendR</code> : Spurious spaces in <code>\pendR</code>	40	<code>\pendL</code> : As in eledmac, <code>\pendL</code> can have an optional argument. . .	39
	<code>\pstartR</code> : Spurious spaces in <code>\pstartL</code> and <code>\pstartR</code>	37	<code>\pendR</code> : As in eledmac, <code>\pendR</code> can have an optional argument. . .	40
v1.5.0	General: Add, as in eledmac, features to manage page breaks. . .	1	<code>\print@columnseparator</code> : Move some code of <code>\Columns</code> to <code>\print@columnseparator</code> . . .	64
	<code>\sublinenumincrement*</code> : Add starred version of <code>\firstlinenum</code> , <code>\linenumincrement</code> , <code>\firstsublinenum</code> , <code>\sublinenumincrement</code> to change both Left and Right-side.	19	<code>\pstartR</code> : As in eledmac, <code>\pendL</code> and <code>\pendR</code> can have an optional argument.	37
v1.6.0	General: Add tool and documentation for parallel ledgroups . . .	10	<code>\sidenotemargin*</code> : <code>\sidenotemargin</code> is now directly defined in eledmac to be able to manage eledpar.	54
v1.7.0	General: Add, as in eledmac, features to make crossrefs with pstart numbers.	1	Add <code>\sidenotemargin*</code>	54
v1.8.0	General: <code>\l@dlsnote</code> , <code>\l@drsnote</code> and <code>\l@dcsnote</code> defined only one time, in eledmac.	54	<code>\theledlanguageR</code> : Correct left/right language setting with polyglossia.	62
	Add <code>\beforecolumnseparator</code> and <code>\aftercolumnseparator</code> . .	4	v1.8.1	
	Add <code>\columnssposition</code>	4	<code>\do@lineL</code> : Fix a bug with critical notes a the begining of a page, (maybe added by v1.8.0) (?). .	41
			<code>\do@lineR</code> : Fix a bug with critical notes a the begining of a page, added by v1.8.0 (?).	42
			v1.8.2	
			General: Debug <code>\eledxxx</code> with some paper size	1
			Debug left and side note (bugs added by 1.8.0)	1
			<code>\eledpar@error</code> : Errors specific to eledpar send to eledpar handbook	14

<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>eled-</code>	<code>par</code> 31
	<code>\lineation*</code> : Add <code>\lineation*</code> . 18