

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.

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.

You can subscribe to the **eledmac** email list in:
<https://lists.berlios.de/pipermail/ledmac-users/>

Contents

1 Introduction	3
2 The eledpar package	4
2.1 General	4
3 Parallel columns	5
4 Facing pages	5
5 Left and right texts	6
6 Numbering text lines and paragraphs	8

*This file (**eledpar.dtx**) has version number v1.5.0, last revised 2013/11/08.

[†]herries dot press at earthlink dot net

[‡]maieul at maieul dot net

7 Verse	9
8 Implementation overview	12
9 Preliminaries	12
9.1 Messages	13
10 Sectioning commands	13
11 Line counting	16
11.1 Choosing the system of lineation	16
11.2 Line-number counters and lists	19
11.3 Reading the line-list file	20
11.4 Commands within the line-list file	21
11.5 Writing to the line-list file	29
12 Marking text for notes	32
13 Parallel environments	33
14 Paragraph decomposition and reassembly	35
14.1 Boxes, counters, <code>\pstart</code> and <code>\pend</code>	36
14.2 Processing one line	39
14.3 Line and page number computation	41
14.4 Line number printing	44
14.5 Pstart number printing in side	46
14.6 Add insertions to the vertical list	47
14.7 Penalties	48
14.8 Printing leftover notes	49
15 Footnotes	49
15.1 Normal footnote formatting	49
16 Cross referencing	50
17 Side notes	51
18 Familiar footnotes	53
19 Verse	54
20 Naming macros	56
21 Counts and boxes for parallel texts	57
22 Fixing babel	58
23 Parallel columns	60

<i>List of Figures</i>	3
24 Parallel pages	64
25 Page break/no page break, depending on the specific line	73
26 The End	74
Appendix A Some things to do when changing version	75
Appendix A.1 Migration to <code>eledpar</code> 1.4.3	75
References	75
Index	75
Change History	84

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 8 through 26); 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 8, unless you are particularly interested in the innards of `eledpar`.

2 The *eledpar* package

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

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

2.1 General

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

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

`\maxchunks`

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

`\maxchunks{5120}`

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

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

(`\Columns` or `\Pages`) more frequently.

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

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

3 Parallel columns

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

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

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

There is no required pagebreak before or after the columns.

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

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

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

\columnrulewidth The macro `\columnseparator` is called between each left/right pair of lines. By default it inserts a vertical rule of width `\columnrulewidth`. As this is initially defined to be 0pt the rule is invisible. For a visible rule between the columns you could try:

```
\setlength{\columnrulewidth}{0.4pt}
```

You can also modify `\columnseparator` if you want more control. When you use `\stanza`, the visible rule may shift when a verse has a hanging indent. To prevent shifting, use `\setstanzaindents` outside the **Leftside** or **Rightside** environment.

4 Facing pages

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

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

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

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

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

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

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

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

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

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

5 Left and right texts

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

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

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

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

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

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

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

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

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

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.

If you are using the `babel` package with different languages (via, say, `\selectlanguage`) for the left and right texts it is particularly important to select the appropriate language within the `Leftside` and `Rightside` environments. The initial language selected for the right text is the `babel` package's default. Also, it is the *last* `\selectlanguage` in a side that controls the language used in any notes for that side when they get printed. If you are using multilingual notes then it is probably safest to explicitly specify the language(s) for each note rather than relying on the language selection for the side. The right side language is also applied to the right side line numbers.

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

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

6 Numbering text lines and paragraphs

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

```
\beginnumbering
<text>
\endnumbering
```

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

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

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

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

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

Its default definition is:

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

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

```
\renewcommand*{\Rlineflag}{}.
```

`\printlinesR`
`\ledsavedprintlines`

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}}
```

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

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`

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 `\setstanzaindent` to set the stanza indents.

`\skipnumbering`

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

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

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

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

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

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

  \end{astanza}
  ...
```

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

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

    \strut &
```

```

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

```

`\hangingsymbol` Like in `eledmac`, you could redefine the command `\hangingsymbol` to insert a character in each 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 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`.

9 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}[2013/11/08 v1.5.0 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 \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
\ifl@dpaging is also set TRUE if we are doing parallel pages. \ifledRcol is set TRUE if we
\ifledRcol are doing the right hand text. \ifl@dpairing is defined in eledmac.
11 \l@dpairingfalse
12 \newif\ifl@dpaging
13 \l@dpagingfalse
14 \ledRcolfalse

\Lcolwidth The widths of the left and right parallel columns (or pages).
\Rcolwidth 15 \newdimen\Lcolwidth
16 \Lcolwidth=0.45\textwidth
17 \newdimen\Rcolwidth
18 \Rcolwidth=0.45\textwidth
19
```

9.1 Messages

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

```
\led@err@TooManyPstarts
20 \newcommand*{\led@err@TooManyPstarts}{%
21 \eledmac@error{Too many \string\pstart\space without printing.
22 Some text will be lost}{\@ehc}}

\led@err@BadLeftRightPstarts
23 \newcommand*{\led@err@BadLeftRightPstarts}[2]{%
24 \eledmac@error{The numbers of left (#1) and right (#2)
25 \string\pstart s do not match}{\@ehc}}

\led@err@LeftOnRightPage
\led@err@RightOnLeftPage 26 \newcommand*{\led@err@LeftOnRightPage}{%
27 \eledmac@error{The left page has ended on a right page}{\@ehc}}
28 \newcommand*{\led@err@RightOnLeftPage}{%
29 \eledmac@error{The right page has ended on a left page}{\@ehc}}
```

10 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.

```
30 \newcount\section@numR
31 \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`.

```

32 \pst@rtedLfalse
33 \newif\ifpst@rtedR
34 \pst@rtedRfalse
35

```

`\beginnumbering` For parallel processing the original `\beginnumbering` is extended to zero `\l@dnumpstartsL` — the number of chunks to be processed. It also sets `\ifpst@rtedL` to FALSE.

```

36 \providecommand*{\beginnumbering}{%
37   \ifnumbering
38     \led@err@NumberingStarted
39   \endnumbering
40   \fi
41   \global\l@dnumpstartsL \z@
42   \global\pst@rtedLfalse
43   \global\numberingtrue
44   \global\advance\section@num \@ne
45   \initnumbering@reg
46   \message{Section \the\section@num}%
47   \line@list@stuff{\jobname.\extensionchars\the\section@num}%
48   \l@dend@stuff}

```

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

```

49 \newcommand*{\beginnumberingR}{%
50   \ifnumberingR
51     \led@err@NumberingStarted
52   \endnumberingR
53   \fi
54   \global\l@dnumpstartsR \z@
55   \global\pst@rtedRfalse
56   \global\numberingRtrue
57   \global\advance\section@numR \@ne
58   \global\absline@numR \z@
59   \gdef\normal@page@breakR{}
60   \gdef\l@prev@pbR{}
61   \gdef\l@prev@nopbR{}
62   \global\line@numR \z@
63   \global\@lockR \z@
64   \global\sub@lockR \z@
65   \global\sublines@false
66   \global\let\next@page@numR\relax
67   \global\let\sub@change\relax
68   \message{Section \the\section@numR R }%
69   \line@list@stuffR{\jobname.\extensionchars\the\section@numR R}%
70   \l@dend@stuff
71   \setcounter{pstartR}{1}
72   \begingroup

```

```

73 \initnumbering@sectcmd
74 \initnumbering@sectcountR
75 }
76

```

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

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

```

77 \def\endnumberingR{%
78   \ifnumberingR
79     \global\numberingRfalse
80     \normal@pars
81     \ifl@dpairing
82       \global\pst@rtedRfalse
83     \else
84       \ifx\insertlines@listR\empty\else
85         \global\noteschanged@true
86       \fi
87       \ifx\line@listR\empty\else
88         \global\noteschanged@true
89       \fi
90     \fi
91     \ifnoteschanged@
92       \led@mess@NotesChanged
93     \fi
94   \else
95     \led@err@NumberingNotStarted
96   \fi\endgroup}
97

```

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

```

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

```

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

```

\resumenumberingR 108 \newcommand*{\pausenumberingR}{%
109   \endnumberingR\global\numberingRtrue}

```

```

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

```

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

```

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

```

11 Line counting

11.1 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.

<pre> \ifbypstart@R \byppstart@Rtrue \byppstart@Rfalse \ifbypage@R \bypage@Rtrue \bypage@Rfalse </pre>	<p>The <code>\ifbypage@R</code> and <code>\ifbypstart@R</code> flag specifie the current lineation system:</p> <ul style="list-style-type: none"> • line-of-page : <code>bypstart@R = false</code> and <code>bypage@R = true</code>. • line-of-pstart : <code>bypstart@R = true</code> and <code>bypage@R = false</code>.
--	---

eledpar will use the line-of-section system unless instructed otherwise.

```
140 \newif\ifbypage@R
141 \newif\ifbypstart@R
142 \bypage@Rfalse
143 \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`.

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

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

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

```
168 \newcount\line@marginR
169 \renewcommand*{\linenummargin}[1]{%
170   \l@getline@margin{#1}%
171   \ifnum\@l@tempcntb>\m@ne
172     \ifledRcol
173       \global\line@marginR=\@l@tempcntb
174     \else
175       \global\line@margin=\@l@tempcntb
```

```

176 \fi
177 \fi}}

```

By default put right text numbers at the right.

```

178 \line@marginR=\@ne
179

```

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

```

180 \newcounter{firstlinenumR}
181 \setcounter{firstlinenumR}{5}
182 \newcounter{linenumincrementR}
183 \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.

```

184 \newcounter{firstsublinenumR}
185 \setcounter{firstsublinenumR}{5}
186 \newcounter{sublinenumincrementR}
187 \setcounter{sublinenumincrementR}{5}
188

```

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

```

\firstlinenum 189 \providecommand*\firstlinenum*{}
\linenumincrement 190 \providecommand*\linenumincrement*{}
\firstsublinenum 191 \providecommand*\firstsublinenum*{}
\sublinenumincrement 192 \providecommand*\sublinenumincrement*{}
\firstlinenum* 193 \renewcommand*\firstlinenum*[1]{%
194 \ifledRcol \setcounter{firstlinenumR}{#1}%
195 \else \setcounter{firstlinenumR}{#1}%
196 \fi}
\linenumincrement* 197 \renewcommand*\linenumincrement*[1]{%
198 \ifledRcol \setcounter{linenumincrementR}{#1}%
199 \else \setcounter{linenumincrementR}{#1}%
200 \fi}
\firstsublinenum* 201 \renewcommand*\firstsublinenum*[1]{%
202 \ifledRcol \setcounter{firstsublinenumR}{#1}%
203 \else \setcounter{firstsublinenumR}{#1}%
204 \fi}
\sublinenumincrement* 205 \renewcommand*\sublinenumincrement*[1]{%
206 \ifledRcol \setcounter{sublinenumincrementR}{#1}%
207 \else \setcounter{sublinenumincrementR}{#1}%
208 \fi}
209 \WithSuffix\newcommand\firstlinenum*[1]{\setcounter{firstlinenumR}{#1}\setcounter{firstlin

```

```

210 \WithSuffix\newcommand\linenumincrement*[1]{\setcounter{linenumincrementR}{#1}\setcounter{linenumincr
211 \WithSuffix\newcommand\firstsublinenum*[1]{\setcounter{subfirstlinenumR}{#1}\setcounter{subfirstlinen
212 \WithSuffix\newcommand\sublinenumincrement*[1]{\setcounter{sublinenumincrementR}{#1}\setcounter{subli

```

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

```

213 \newcommand*{\Rlineflag}{R}
214

```

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

```

215 \newcommand*{\linenumrepR}[1]{\@arabic{#1}}
216 \newcommand*{\sublinenumrepR}[1]{\@arabic{#1}}
217

```

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

```

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

```

11.2 Line-number counters and lists

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

`\line@numR` The count `\line@numR` stores the line number that's used in the right text's
`\subline@numR` marginal line numbering and in notes. The count `\subline@numR` stores a sub-line
`\absline@numR` 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.

```

232 \newcount\line@numR
233 \newcount\subline@numR
234 \newcount\absline@numR
235

```

`\line@listR` Now we can define the list macros that will be created from the line-list file. They
`\insertlines@listR` are directly analagous to the left text ones. The full list of action codes and their
`\actionlines@listR` meanings is given in the `eledmac` manual.
`\actions@listR` Here are the commands to create these lists:

```
236 \list@create{\line@listR}
237 \list@create{\insertlines@listR}
238 \list@create{\actionlines@listR}
239 \list@create{\actions@listR}
240
```

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

```
241 \list@create{\linesinpar@listL}
242 \list@create{\linesinpar@listR}
243 \list@create{\maxlinesinpar@list}
244
```

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

```
245 \newcount\page@numR
246
```

11.3 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.

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

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

```
248 \ifledRcol
249 \list@clear{\line@listR}%
250 \list@clear{\insertlines@listR}%
251 \list@clear{\actionlines@listR}%
252 \list@clear{\actions@listR}%
253 \list@clear{\linesinpar@listR}%
254 \list@clear{\linesonpage@listR}
255 \else
256 \list@clearing@reg
257 \list@clear{\linesinpar@listL}%
258 \list@clear{\linesonpage@listL}%
259 \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).

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

Now get the file and interpret it.

```
261 \get@linelistfile{#1}%
262 \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.

```

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

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.

11.4 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.

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

```

281 \newcommand{\@l@regR}{%
282   \ifx\l@dchset@num\relax \else
283     \advance\absline@numR \@ne
284     \set@line@action
285     \let\l@dchset@num\relax
286     \advance\absline@numR \m@ne
287     \advance\line@numR \m@ne%    % do we need this?
288   \fi
289   \advance\absline@numR \@ne
290   \ifx\next@page@numR\relax \else
291     \page@action

```

```

292 \let\next@page@numR\relax
293 \fi
294 \ifx\sub@change\relax \else
295 \ifnum\sub@change>\z@
296 \sublines@true
297 \else
298 \sublines@false
299 \fi
300 \sub@action
301 \let\sub@change\relax
302 \fi
303 \ifcase\@lockR
304 \or
305 \@lockR \tw@
306 \or\or
307 \@lockR \z@
308 \fi
309 \ifcase\sub@lockR
310 \or
311 \sub@lockR \tw@
312 \or\or
313 \sub@lockR \z@
314 \fi
315 \ifsublines@
316 \ifnum\sub@lockR<\tw@
317 \advance\subline@numR \@ne
318 \fi
319 \else
320 \ifnum\@lockR<\tw@
321 \advance\line@numR \@ne \subline@numR \z@
322 \fi
323 \fi}
324
325 \renewcommand*{\@l}[2]{%
326 \fix@page{#1}%
327 \ifledRcol
328 \@l@regR
329 \else
330 \@l@reg
331 \fi}
332

```

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

```

\fix@page 333 \newcount\last@page@numR
334 \last@page@numR=-10000
335 \renewcommand*{\fix@page}[1]{%
336 \ifledRcol
337 \ifnum #1=\last@page@numR
338 \else
339 \ifbypage@R

```

```

340     \line@numR \z@ \subline@numR \z@
341     \fi
342     \page@numR=#1\relax
343     \last@page@numR=#1\relax
344     \def\next@page@numR{#1}%
345     \fi
346 \else
347     \ifnum #1=\last@page@num
348     \else
349         \ifbypage@
350             \line@num \z@ \subline@num \z@
351             \fi
352             \page@num=#1\relax
353             \last@page@num=#1\relax
354             \def\next@page@num{#1}%
355             \listcsxadd{normal@page@break}{\the\absline@num}
356         \fi
357     \fi}
358

```

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

```

359 \renewcommand*{\@adv}[1]{%
360     \ifsublines@
361     \ifledRcol
362         \advance\subline@numR by #1\relax
363         \ifnum\subline@numR<\z@
364             \led@warn@BadAdvancelineSubline
365             \subline@numR \z@
366         \fi
367     \else
368         \advance\subline@num by #1\relax
369         \ifnum\subline@num<\z@
370             \led@warn@BadAdvancelineSubline
371             \subline@num \z@
372         \fi
373     \fi
374 \else
375     \ifledRcol
376         \advance\line@numR by #1\relax
377         \ifnum\line@numR<\z@
378             \led@warn@BadAdvancelineLine
379             \line@numR \z@
380         \fi
381     \else
382         \advance\line@num by #1\relax
383         \ifnum\line@num<\z@
384             \led@warn@BadAdvancelineLine
385             \line@num \z@
386         \fi

```

```

387   \fi
388   \fi
389   \set@line@action}
390

```

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

```

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

```

`\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 linenummer change is to be done.

```

408 \renewcommand*{\l@d@set}[1]{%
409   \ifledRcol
410     \line@numR=#1\relax
411     \advance\line@numR \@ne
412     \def\l@dchset@num{#1}
413   \else
414     \line@num=#1\relax
415     \advance\line@num \@ne
416     \def\l@dchset@num{#1}
417   \fi}
418 \let\l@dchset@num\relax
419

```

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

```

420 \renewcommand*{\page@action}{%
421   \ifledRcol
422     \xright@appenditem{\the\absline@numR}\to\actionlines@listR
423     \xright@appenditem{\next@page@numR}\to\actions@listR
424   \else
425     \xright@appenditem{\the\absline@num}\to\actionlines@list

```

```

426   \xright@appenditem{\next@page@num}\to\actions@list
427   \fi}

```

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

```

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

```

\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.

```

449 \renewcommand*{\sub@action}{%
450   \ifledRcol
451     \xright@appenditem{\the\absline@numR}\to\actionlines@listR
452     \ifsublines@
453       \xright@appenditem{-1001}\to\actions@listR
454     \else
455       \xright@appenditem{-1002}\to\actions@listR
456     \fi
457   \else
458     \xright@appenditem{\the\absline@num}\to\actionlines@list
459     \ifsublines@
460       \xright@appenditem{-1001}\to\actions@list
461     \else
462       \xright@appenditem{-1002}\to\actions@list
463     \fi
464   \fi}
465

```

\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.

```

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

```

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

```

\do@lockoff 501
\do@lockoffR 502
\skip@lockoff 503 \newcommand*\do@lockoffR}{%
504   \xright@appenditem{\the\absline@numR}\to\actionlines@listR
505   \ifsublines@
506     \xright@appenditem{-1006}\to\actions@listR
507     \ifnum\sub@lockR=\tw@
508       \sub@lockR \thr@@
509     \else
510       \sub@lockR \z@
511     \fi
512   \else
513     \xright@appenditem{-1004}\to\actions@listR

```

```

514 \ifnum\@lockR=\tw@
515 \@lockR \thr@@
516 \else
517 \@lockR \z@
518 \fi
519 \fi}
520
521 \renewcommand*{\do@lockoff}{%
522 \ifledRcol
523 \do@lockoffR
524 \else
525 \do@lockoffL
526 \fi}
527 \global\let\lock@off=\do@lockoff
528

```

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

```

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

```

`\@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`.

```

538 \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.

```

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

```

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

```

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

```

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

```

562 #2
563 \else

```

And when not in right text

```

564   \@ref@reg{#1}{#2}%
565 \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.

```

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

```

`\@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.

```

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

```

11.5 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`.

```

582 \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
583 \newif\iffirst@linenum@out@R
584 \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.

```

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

```

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

```

606 \newcommand*{\new@lineL}{%
607   \write\linenum@out{\string\@l[\the\c@page][\thepage]}}
```

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

```

608 \newcommand*{\new@lineR}{%
609   \write\linenum@outR{\string\@r[\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.

```

610 \renewcommand*{\flag@start}{%
611   \ifledRcol
612     \edef\next{\write\linenum@outR{%
613       \string\@ref[\the\insert@countR] []}}%
614   \next
615 \else
616   \edef\next{\write\linenum@out{%
617     \string\@ref[\the\insert@count] []}}%
618   \next
619 \fi}
620 \renewcommand*{\flag@end}{%
621   \ifledRcol
622     \write\linenum@outR{[]}%
623   \else
624     \write\linenum@out{[]}%
625   \fi}
```

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

```

626 \renewcommand*{\startsub}{\dimen0\lastskip
627   \ifdim\dimen0>0pt \unskip \fi
628   \ifledRcol \write\linenum@outR{\string\sub@on}%
629   \else      \write\linenum@out{\string\sub@on}%
630   \fi
631   \ifdim\dimen0>0pt \hskip\dimen0 \fi}
632 \def\endsub{\dimen0\lastskip
633   \ifdim\dimen0>0pt \unskip \fi
634   \ifledRcol \write\linenum@outR{\string\sub@off}%
635   \else      \write\linenum@out{\string\sub@off}%
636   \fi
637   \ifdim\dimen0>0pt \hskip\dimen0 \fi}
638
```

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

```

639 \renewcommand*{\advanceline}[1]{%
640   \ifledRcol \write\linenum@outR{\string\@adv[#1]}%
```

```

641 \else      \write\linenum@out{\string\@adv[#1]}%
642 \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.

```

643 \renewcommand*{\setline}[1]{%
644 \ifnum#1<\z@
645 \led@warn@BadSetline
646 \else
647 \ifledRcol \write\linenum@outR{\string\@set[#1]}%
648 \else      \write\linenum@out{\string\@set[#1]}%
649 \fi
650 \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.

```

651 \renewcommand*{\setlinenum}[1]{%
652 \ifnum#1<\z@
653 \led@warn@BadSetlinenum
654 \else
655 \ifledRcol \write\linenum@outR{\string\l@d@set[#1]}
656 \else      \write\linenum@out{\string\l@d@set[#1]} \fi
657 \fi}
658

```

\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.

```

659 \renewcommand*{\startlock}{%
660 \ifledRcol \write\linenum@outR{\string\lock@on}%
661 \else      \write\linenum@out{\string\lock@on}%
662 \fi}
663 \def\endlock{%
664 \ifledRcol \write\linenum@outR{\string\lock@off}%
665 \else      \write\linenum@out{\string\lock@off}%
666 \fi}
667

```

\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.

```

668 \renewcommand*{\skipnumbering}{%
669 \ifledRcol \write\linenum@outR{\string\n@num}%
670 \advanceline{-1}%
671 \else
672 \skipnumbering@reg
673 \fi}
674

```

12 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.

```
675 \long\def\critext#1#2/{\leavevmode
676   \begingroup
677     \renewcommand{\@tag}{\no@expands #1}%
678     \set@line
679     \ifl@Rcol \global\insert@countR \z@
680     \else      \global\insert@count \z@ \fi
681     \ignorespaces #2\relax
682     \@ifundefined{xpg@main@language}{%if not polyglossia
683       \flag@start}%
684       {\if@RTL\flag@end\else\flag@start\fi% be careful on the direction of writing with p
685     }%
686   \endgroup
687   \showlemma{#1}%
688   \ifx\end@lemmas\empty \else
689     \gl@p\end@lemmas\to\x@lemma
690     \x@lemma
691     \global\let\x@lemma=\relax
692   \fi
693   \@ifundefined{xpg@main@language}{%if not polyglossia
694     \flag@end}%
695     {\if@RTL\flag@start\else\flag@end\fi% be careful on the direction of writing with p
696   }
697 }
```

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

```
698 \renewcommand{\edtext}[2]{\leavevmode
699   \begingroup%
700     \renewcommand{\@tag}{\no@expands #1}%
701     \set@line%
```

```

702 \ifledRcol \global\insert@countR \z@%
703 \else \global\insert@count \z@ \fi%
704 \ignorespaces #2\relax%
705 \@ifundefined{xpg@main@language}{%if not polyglossia
706 \flag@start}%
707 {\if@RTL\flag@end\else\flag@start\fi% be careful on the direction of writing with polyglossia
708 }%
709 \endgroup%
710 \showlemma{#1}%
711 \ifx@end@lemmas\empty \else%
712 \gl@p@end@lemmas\to\x@lemma%
713 \x@lemma%
714 \global\let\x@lemma=\relax%
715 \fi%
716 \@ifundefined{xpg@main@language}{%if not polyglossia
717 \flag@end}%
718 {\if@RTL\flag@start\else\flag@end\fi% be careful on the direction of writing with polyglossia
719 }%
720 }
721

```

`\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`.

```

722 \renewcommand*{\set@line}{%
723 \ifledRcol
724 \ifx\line@listR\empty
725 \global\noteschanged@true
726 \xdef\l@d@nums{000|000|000|000|000|000|\edfont@info}%
727 \else
728 \gl@p\line@listR\to\@tempb
729 \xdef\l@d@nums{\@tempb|\edfont@info}%
730 \global\let\@tempb=\undefined
731 \fi
732 \else
733 \ifx\line@list\empty
734 \global\noteschanged@true
735 \xdef\l@d@nums{000|000|000|000|000|000|\edfont@info}%
736 \else
737 \gl@p\line@list\to\@tempb
738 \xdef\l@d@nums{\@tempb|\edfont@info}%
739 \global\let\@tempb=\undefined
740 \fi
741 \fi}
742

```

13 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

```

743 \newenvironment{pairs}{%
744   \l@dpairingtrue
745   \l@dpagingfalse
746 }{%
747   \l@dpairingfalse
748 }

```

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.

```

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

```

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

ifinstanzaR

```

766 \newif\ifinstanzaL
767 \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.

```

768 \newenvironment{Leftside}{%
769   \ledRcolfalse
770   \let\beginnumbering\beginnumbering\setcounter{pstartL}{1}
771   \let\pstart\pstartL
772   \let\thepstart\thepstartL
773   \let\pend\pendL
774   \let\memorydump\memorydumpL
775   \Leftsidehook
776   \let\oldstanza\stanza

```

```

777 \renewcommand{\stanza}{\oldstanza\global\istanzaLtrue}
778 }{
779 \let\stanza\oldstanza
780 \Leftsidehookend}

```

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

```

\Rightsidehook 781 \newcommand*{\Leftsidehook}{}
\Rightsidehookend 782 \newcommand*{\Leftsidehookend}{}
783 \newcommand*{\Rightsidehook}{}
784 \newcommand*{\Rightsidehookend}{}
785

```

`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.

```

786 \newenvironment{Rightside}{%
787 \ledRcoltrue
788 \let\beginnumbering\beginnumberingR
789 \let\endnumbering\endnumberingR
790 \let\pausenumbering\pausenumberingR
791 \let\resumenumbering\resumenumberingR
792 \let\memorydump\memorydumpR
793 \let\thepstart\thepstartR
794 \let\pstart\pstartR
795 \let\pend\pendR
796 \let\ledpb\ledpbR
797 \let\lednopb\lednopbR
798 \let\lineation\lineationR
799 \Rightsidehook
800 \let\oldstanza\stanza
801 \renewcommand{\stanza}{\oldstanza\global\istanzaRtrue}
802 }{%
803 \ledRcolfalse
804 \let\stanza\oldstanza
805 \Rightsidehookend
806 }
807

```

14 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.

14.1 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.

```
808 \newcount\num@linesR
```

```
809 \newbox\one@lineR
```

```
810 \newcount\par@lineR
```

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

`\pstartR`

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

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

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

833 \fi%

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

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

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

860 \newcommand*{\pstartR}{%
861 \if@nobreak%
862   \let\@oldnobreak\@nobreaktrue%
863 \else%
864   \let\@oldnobreak\@nobreakfalse%
865 \fi%
866   \@nobreaktrue%
867   \ifnumberingR \else%
868     \led@err@PstartNotNumbered%
869     \beginnumberingR%
870   \fi%
871   \ifnumberedpar@%
872     \led@err@PstartInPstart%
873   \pendR%
874 \fi%
875 \ifpst@rtedR\else%
876   \setcounter{pstartRold}{\value{pstartR}}%
```

```

877 \list@clear{\inserts@listR}%
878 \global\let\next@insertR=\empty%
879 \global\pst@rtedRtrue%
880 \fi%
881 \begingroup\normal@pars%
882 \global\advance\l@dnumpstartsR \@ne%
883 \ifnum\l@dnumpstartsR>\l@dc@maxchunks%
884 \led@err@TooManyPstarts%
885 \global\l@dnumpstartsR=\l@dc@maxchunks%
886 \fi%
887 \global\setnamebox{\l@dr@colrawbox\the\l@dnumpstartsR}=\vbox\bgroup%
888 \ifautopar\else%
889 \ifnumberpstart%
890 \ifsidepstartnum\else%
891 \thepstartR%
892 \fi%
893 \fi%
894 \fi%
895 \hsize=\Rcolwidth%
896 \numberedpar@true%
897 \iflabelpstart\protected@edef\@currentlabel%
898 {\p@pstartR\thepstartR}\fi%
899 }

```

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

```

900 \newcommand*{\pendL}{\ifnumbering \else%
901 \led@err@PendNotNumbered%
902 \fi%
903 \ifnumberedpar@ \else%
904 \led@err@PendNoPstart%
905 \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.

```

906 \l@dzeropenalties%
907 \endgraf\global\num@lines=\prevgraf\egroup%
908 \global\par@line=0%

```

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

```

909 \endgroup%
910 \ignorespaces%
911 \@oldnobreak%
912 \ifnumberpstart%
913 \addtocounter{pstartL}{1}%
914 \fi%
915

```

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

```

916 \newcommand*{\pendR}{\ifnumberingR \else%
917   \led@err@PendNotNumbered%
918 \fi%
919 \ifnumberedpar@ \else%
920   \led@err@PendNoPstart%
921 \fi%
922 \l@dzeropenalties%
923 \endgraf\global\num@linesR=\prevgraf\egroup%
924 \global\par@lineR=0%
925 \endgroup%
926 \ignorespaces%
927 \@oldnobreak%
928   \ifnumberpstart%
929   \addtocounter{pstartR}{1}%
930 \fi%
931 }
932
```

14.2 Processing one line

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

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

```

933 \newbox\l@dleftbox
934 \newbox\l@drightbox
935
```

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

```

\countRline 936 \newcount\countLline
937   \countLline \z@
938 \newcount\countRline
939   \countRline \z@
940
```

`\@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 941 \newcount\@donereallinesL
942 \newcount\@donetotallinesL
943 \newcount\@donereallinesR
944 \newcount\@donetotallinesR
945
```

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

```

946 \newcommand*{\do@lineL}{%
947   \advance\countLline \@ne
948   \ifvbox\namebox{\l@dLcolrawbox\the\l@dpscL}%
949   {\vbadness=10000
950    \splittopskip=\z@
951    \do@lineLhook
952    \l@emptyd@ta
953    \global\setbox\one@line=\vsplit\namebox{\l@dLcolrawbox\the\l@dpscL}
954      to\baselineskip}%
955   \unvbox\one@line \global\setbox\one@line=\lastbox
956   \getline@numL
957   \ifnum\@lock>\@ne\inserthangingsymboltrue\else\inserthangingsymbolfalse\fi
958   \setbox\l@dleftbox
959   \hb@xt@ \Lcolwidth{%
960     \affixstart@numL
961     \affixline@num
962     \l@dld@ta
963     \add@inserts
964     \affixside@note
965     \l@dlsn@te
966     {\ledllfill\hb@xt@ \wd\one@line{\do@insidelineLhook\inserthangingsymbolL\new@lineL\l@
967       \l@drsn@te
968     }}%
969   \add@penaltiesL
970   \global\advance\@donereallinesL\@ne
971   \global\advance\@donetotallinesL\@ne
972 \else
973   \setbox\l@dleftbox \hb@xt@ \Lcolwidth{\hspace*\Lcolwidth}}}%
974   \global\advance\@donetotallinesL\@ne
975 \fi}
976
977

```

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

```

\do@lineRhook 978 \newcommand*{\do@lineLhook}{}
\do@insidelineLhook 979 \newcommand*{\do@lineRhook}{}
\do@insidelineRhook 980 \newcommand*{\do@insidelineLhook}{}
981 \newcommand*{\do@insidelineRhook}{}
982

```

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

```

983 \newcommand*{\do@lineR}{%
984   \advance\countRline \@ne
985   \ifvbox\namebox{\l@dRcolrawbox\the\l@dpscR}%
986   {\vbadness=10000
987    \splittopskip=\z@
988    \do@lineRhook
989    \l@emptyd@ta

```

```

990 \global\setbox\one@lineR=\vsplit\namebox{1@dRcolrawbox\the\l@dpscR}
991 to\baselineskip}%
992 \unvbox\one@lineR \global\setbox\one@lineR=\lastbox
993 \getline@numR
994 \ifnum\@lockR>\@one\inserthangingsymbolRtrue\else\inserthangingsymbolRfalse\fi
995 \setbox\l@drightbox
996 \hb@xt@ \Rcolwidth{%
997 \affixpstart@numR
998 \affixline@numR
999 \l@dld@ta
1000 \add@insertsR
1001 \affixside@noteR
1002 \l@dlsn@te
1003 {\correcthangingsR\ledllfill\hb@xt@ \wd\one@lineR{\do@insidelineRhook\inserthangingsymbolR\new@lin
1004 \l@drsn@te
1005 }}%
1006 \add@penaltiesR
1007 \global\advance\@donereallinesR\@one
1008 \global\advance\@donetotallinesR\@one
1009 \else
1010 \setbox\l@drightbox \hb@xt@ \Rcolwidth{\hspace*{\Rcolwidth}}
1011 \global\advance\@donetotallinesR\@one
1012 \fi}
1013
1014

```

14.3 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.

```

1015 \newcommand*{\getline@numR}{%
1016 \global\advance\absline@numR \@one
1017 \do@actionsR
1018 \do@ballastR
1019 \ifnumberline
1020 \ifsublines@
1021 \ifnum\sub@lockR<\tw@
1022 \global\advance\subline@numR \@one
1023 \fi
1024 \else
1025 \ifnum\@lockR<\tw@
1026 \global\advance\line@numR \@one
1027 \global\subline@numR \z@
1028 \fi
1029 \fi
1030 \fi
1031 }
1032 \newcommand*{\getline@numL}{%
1033 \global\advance\absline@num \@one

```

```

1034 \do@actions
1035 \do@ballast
1036 \ifnumberline
1037 \ifsublines@
1038   \ifnum\sub@lock<\tw@
1039     \global\advance\subline@num \@ne
1040   \fi
1041 \else
1042   \ifnum\@lock<\tw@
1043     \global\advance\line@num \@ne
1044     \global\subline@num \z@
1045   \fi
1046 \fi
1047 \fi
1048 }
1049
1050

```

`\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.

```

1051 \newcommand*{\do@ballastR}{\global\ballast@count=\z@
1052 \begingroup
1053   \advance\absline@numR \@ne
1054   \ifnum\next@actionlineR=\absline@numR
1055     \ifnum\next@actionR>-1001
1056       \global\advance\ballast@count by -\c@ballast
1057     \fi
1058   \fi
1059 \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.

```

1060 \newcommand*{\do@actions@fixedcodeR}{%
1061   \ifcase\@l@dttempcnta%
1062     \or% % 1001
1063       \global\sublines@true
1064     \or% % 1002
1065       \global\sublines@false
1066     \or% % 1003
1067       \global\@lockR=\@ne
1068     \or% % 1004
1069       \ifnum\@lockR=\tw@
1070         \global\@lockR=\thr@@
1071       \else
1072         \global\@lockR=\z@
1073       \fi

```

```

1074 \or% % 1005
1075 \global\sub@lockR=\@ne
1076 \or% % 1006
1077 \ifnum\sub@lockR=\tw@
1078 \global\sub@lockR=\thr@@
1079 \else
1080 \global\sub@lockR=\z@
1081 \fi
1082 \or% % 1007
1083 \l@dskipnumbertrue
1084 \else
1085 \led@warn@BadAction
1086 \fi}
1087
1088
1089 \newcommand*{\do@actionsR}{%
1090 \global\let\do@actions@nextR=\relax
1091 \@l@dttempcntb=\absline@numR
1092 \ifnum\@l@dttempcntb<\next@actionlineR\else
1093 \ifnum\next@actionR>-1001\relax
1094 \global\page@numR=\next@actionR
1095 \ifbypage@R
1096 \global\line@numR \z@ \global\subline@numR \z@
1097 \fi
1098 \else
1099 \ifnum\next@actionR<-4999\relax % 9/05 added relax here
1100 \@l@dttempcnta=-\next@actionR
1101 \advance\@l@dttempcnta by -5001\relax
1102 \ifsublines@
1103 \global\subline@numR=\@l@dttempcnta
1104 \else
1105 \global\line@numR=\@l@dttempcnta
1106 \fi
1107 \else
1108 \@l@dttempcnta=-\next@actionR
1109 \advance\@l@dttempcnta by -1000\relax
1110 \do@actions@fixedcodeR
1111 \fi
1112 \fi
1113 \ifx\actionlines@listR\empty
1114 \gdef\next@actionlineR{1000000}%
1115 \else
1116 \gl@p\actionlines@listR\to\next@actionlineR
1117 \gl@p\actions@listR\to\next@actionR
1118 \global\let\do@actions@nextR=\do@actionsR
1119 \fi
1120 \fi
1121 \do@actions@nextR}
1122

```

14.4 Line number printing

```

\l@dcalcnun \affixline@numR is the right text version of the \affixline@num macro.
\ch@cksub@l@ckR 1123
\ch@ck@l@ckR 1124 \providecommand*\l@dcalcnun}[3]{%
\fx@l@cksR 1125 \ifnum #1 > #2\relax
\affixline@numR 1126 \@l@tempcnta = #1\relax
1127 \advance\@l@tempcnta by -#2\relax
1128 \divide\@l@tempcnta by #3\relax
1129 \multiply\@l@tempcnta by #3\relax
1130 \advance\@l@tempcnta by #2\relax
1131 \else
1132 \@l@tempcnta=#2\relax
1133 \fi}
1134
1135 \newcommand*\ch@cksub@l@ckR}{%
1136 \ifcase\sub@lockR
1137 \or
1138 \ifnum\sublock@disp=\@ne
1139 \@l@tempcntb \z@ \@l@tempcnta \@ne
1140 \fi
1141 \or
1142 \ifnum\sublock@disp=\tw@
1143 \else
1144 \@l@tempcntb \z@ \@l@tempcnta \@ne
1145 \fi
1146 \or
1147 \ifnum\sublock@disp=\z@
1148 \@l@tempcntb \z@ \@l@tempcnta \@ne
1149 \fi
1150 \fi}
1151
1152 \newcommand*\ch@ck@l@ckR}{%
1153 \ifcase\@lockR
1154 \or
1155 \ifnum\lock@disp=\@ne
1156 \@l@tempcntb \z@ \@l@tempcnta \@ne
1157 \fi
1158 \or
1159 \ifnum\lock@disp=\tw@
1160 \else
1161 \@l@tempcntb \z@ \@l@tempcnta \@ne
1162 \fi
1163 \or
1164 \ifnum\lock@disp=\z@
1165 \@l@tempcntb \z@ \@l@tempcnta \@ne
1166 \fi
1167 \fi}
1168
1169 \newcommand*\fx@l@cksR}{%

```

```

1170 \ifcase\@lockR
1171 \or
1172   \global\@lockR \tw@
1173 \or \or
1174   \global\@lockR \z@
1175 \fi
1176 \ifcase\sub@lockR
1177 \or
1178   \global\sub@lockR \tw@
1179 \or \or
1180   \global\sub@lockR \z@
1181 \fi}
1182
1183
1184 \newcommand*{\affixline@numR}{%
1185 \ifnumberline
1186 \ifl@dskipnumber
1187   \global\l@dskipnumberfalse
1188 \else
1189   \ifsublines@
1190     \@l@tempcntb=\subline@numR
1191     \l@dcalcnnum{\subline@numR}{\c@firstsublinenumR}{\c@sublinenumincrementR}%
1192     \ch@cksub@lockR
1193   \else
1194     \@l@tempcntb=\line@numR
1195     \ifx\linenumberlist\empty
1196       \l@dcalcnnum{\line@numR}{\c@firstlinenumR}{\c@linenumincrementR}%
1197     \else
1198       \@l@tempcnta=\line@numR
1199       \edef\rem@inder{\linenumberlist,\number\line@numR,}%
1200       \edef\sc@n@list{\def\noexpand\sc@n@list
1201         ###1,\number\@l@tempcnta,###2|{\def\noexpand\rem@inder{###2}}}%
1202       \sc@n@list\expandafter\sc@n@list\rem@inder|
1203       \ifx\rem@inder\empty\advance\@l@tempcnta\@ne\fi
1204     \fi
1205     \ch@ck@l@ckR
1206   \fi
1207   \ifnum\@l@tempcnta=\@l@tempcntb
1208     \if@twocolumn
1209       \if@firstcolumn
1210         \gdef\l@dld@ta{\llap{\leftlinenumR}}}%
1211       \else
1212         \gdef\l@drd@ta{\rlap{\rightlinenumR}}}%
1213     \fi
1214   \else
1215     \@l@tempcntb=\line@marginR
1216     \ifnum\@l@tempcntb>\@ne
1217       \advance\@l@tempcntb by\page@numR
1218     \fi
1219     \ifodd\@l@tempcntb

```

```

1220      \gdef\l@drd@ta{\rlap{{\rightlinenumR}}}%
1221      \else
1222      \gdef\l@dld@ta{\llap{{\leftlinenumR}}}%
1223      \fi
1224  \fi
1225  \fi
1226  \f@x@l@cksR
1227 \fi
1228 \fi}

```

14.5 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 1229
  \leftpstartnumR 1230 \newcommand*{\affixpstart@numL}{%
\rightpstartnumR 1231 \ifsidepstartnum
  \leftpstartnumL 1232 \if@twocolumn
\rightpstartnumL 1233   \if@firstcolumn
  \ifpstartnumR 1234     \gdef\l@dld@ta{\llap{{\leftpstartnumL}}}%
1235     \else
1236     \gdef\l@drd@ta{\rlap{{\rightpstartnumL}}}%
1237     \fi
1238   \else
1239     \@l@tempcntb=\line@margin
1240     \ifnum\@l@tempcntb>\@ne
1241       \advance\@l@tempcntb \page@num
1242     \fi
1243     \ifodd\@l@tempcntb
1244       \gdef\l@drd@ta{\rlap{{\rightpstartnumL}}}%
1245     \else
1246       \gdef\l@dld@ta{\llap{{\leftpstartnumL}}}%
1247     \fi
1248   \fi
1249 \fi
1250 }
1251 \newcommand*{\affixpstart@numR}{%
1252 \ifsidepstartnum
1253 \if@twocolumn
1254   \if@firstcolumn
1255     \gdef\l@dld@ta{\llap{{\leftpstartnumR}}}%
1256   \else
1257     \gdef\l@drd@ta{\rlap{{\rightpstartnumR}}}%

```

```

1258     \fi
1259   \else
1260     \@l@dttempcntb=\line@marginR
1261     \ifnum\@l@dttempcntb>\@ne
1262       \advance\@l@dttempcntb \page@numR
1263     \fi
1264     \ifodd\@l@dttempcntb
1265       \gdef\l@drd@ta{\rlap{\{\rightpstartnumR\}}}%
1266     \else
1267       \gdef\l@dld@ta{\llap{\{\leftpstartnumR\}}}%
1268     \fi
1269   \fi
1270 \fi
1271 }
1272
1273 \newcommand*{\leftpstartnumL}{
1274 \ifpstartnum
1275 \thepstartL
1276 \kern\linenumsep\global\pstartnumfalse\fi
1277 }
1278 \newcommand*{\rightpstartnumL}{
1279 \ifpstartnum\kern\linenumsep
1280 \thepstartL
1281 \global\pstartnumfalse\fi
1282 }
1283 \newif\ifpstartnumR
1284 \pstartnumRtrue
1285 \newcommand*{\leftpstartnumR}{
1286 \ifpstartnumR
1287 \thepstartR
1288 \kern\linenumsep\global\pstartnumRfalse\fi
1289 }
1290 \newcommand*{\rightpstartnumR}{
1291 \ifpstartnumR\kern\linenumsep
1292 \thepstartR
1293 \global\pstartnumRfalse\fi
1294 }

```

14.6 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.

```
1295 \list@create{\inserts@listR}
```

`\add@insertsR` The right text version.

```

\add@inserts@nextR 1296 \newcommand*{\add@insertsR}{%
1297   \global\let\add@inserts@nextR=\relax
1298   \ifx\inserts@listR\empty \else
1299     \ifx\next@insertR\empty

```

```

1300     \ifx\insertlines@listR\empty
1301         \global\noteschanged@true
1302         \gdef\next@insertR{100000}%
1303     \else
1304         \gl@p\insertlines@listR\to\next@insertR
1305     \fi
1306 \fi
1307 \ifnum\next@insertR=\absline@numR
1308     \gl@p\inserts@listR\to\@insertR
1309     \@insertR
1310     \global\let\@insertR=\undefined
1311     \global\let\next@insertR=\empty
1312     \global\let\add@inserts@nextR=\add@insertsR
1313 \fi
1314 \fi
1315 \add@inserts@nextR}
1316

```

14.7 Penalties

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

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

```

\newcommand*{\add@penaltiesR}{\@l@dttempcnta=\ballast@count
\ifnum\num@linesR>\@ne
\global\advance\par@lineR \@ne
\ifnum\par@lineR=\@ne
\advance\@l@dttempcnta by \clubpenalty
\fi
\@l@dttempcntb=\par@lineR \advance\@l@dttempcntb \@ne
\ifnum\@l@dttempcntb=num@linesR
\advance\@l@dttempcnta by \widowpenalty
\fi
\ifnum\par@lineR<\num@linesR
\advance\@l@dttempcnta by \interlinepenalty
\fi
\fi
\ifnum\@l@dttempcnta=\z@
\relax
\else

```

```

\ifnum\l@dttempcnta>-10000
\penalty\l@dttempcnta
\else
\penalty -10000
\fi
\fi}

```

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

```

1317 \newcommand*{\add@penaltiesL}{\fi}
1318 \newcommand*{\add@penaltiesR}{\fi}
1319

```

14.8 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.

```

1320 \newcommand*{\flush@notesR}{%
1321 \xloop
1322 \ifx\inserts@listR\empty \else
1323 \glp\inserts@listR\to\@insertR
1324 \@insertR
1325 \global\let\@insertR=\undefined
1326 \repeat}
1327

```

15 Footnotes

15.1 Normal footnote formatting

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

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

Just a reminder of the arguments:

```

\printlinesR #1 | #2 | #3 | #4 | #5 | #6 | #7
\printlinesR start-page | line | subline | end-page | line | subline | font

```

```

1328 \def\printlinesR#1|#2|#3|#4|#5|#6|#7|{\begingroup
1329 \setprintlines{#1}{#2}{#3}{#4}{#5}{#6}%
1330 \ifl@d@pnum #1\fullstop\fi

```

```

1331 \ifledplinum \linenumr@p{#2}\Rlineflag\else \symplinum\fi
1332 \ifl@d@ssub \fullstop \sublinenumr@p{#3}\fi
1333 \ifl@d@dash \endashchar\fi
1334 \ifl@d@pnum #4\fullstop\fi
1335 \ifl@d@elin \linenumr@p{#5}\Rlineflag\fi
1336 \ifl@d@esl \ifl@d@elin \fullstop\fi \sublinenumr@p{#6}\fi
1337 \endgroup}
1338
1339 \let\ledsavedprintlines\printlines
1340

```

16 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.

```

1341 \list@create{\labelref@listR}
1342

```

`\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.

```

1343 \renewcommand*{\edlabel}[1]{\@bsphack
1344   \ifledRcol
1345     \write\linenum@outR{\string\@lab}%
1346     \ifx\labelref@listR\empty
1347       \xdef\label@refs{\zz@@@}%
1348     \else
1349       \glp\labelref@listR\to\label@refs
1350     \fi
1351     \ifvmode
1352       \advancelabel@refs
1353     \fi
1354     \protected@write\@auxout{%
1355       {\string\l@dmake@labelsR\space\thepage|\label@refs|{#1}}}%
1356   \else
1357     \write\linenum@out{\string\@lab}%
1358     \ifx\labelref@list\empty
1359       \xdef\label@refs{\zz@@@}%
1360     \else
1361       \glp\labelref@list\to\label@refs
1362     \fi
1363     \ifvmode
1364       \advancelabel@refs
1365     \fi
1366     \protected@write\@auxout{%
1367       {\string\l@dmake@labels\space\thepage|\label@refs|{#1}}}%

```

```

1368 \fi
1369 \@esphack}
1370

```

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

```

1371 \def\l@dmake@labelsR#1|#2|#3|#4{%
1372   \expandafter\ifx\csname the@label#4\endcsname \relax\else
1373     \led@warn@DuplicateLabel{#4}%
1374   \fi
1375   \expandafter\gdef\csname the@label#4\endcsname{#1|#2\Rlineflag|#3}%
1376   \ignorespaces}
1377 \AtBeginDocument{%
1378   \def\l@dmake@labelsR#1|#2|#3|#4{%
1379 }
1380

```

`\@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`, `\@l`, and the `\sub@on` and `\sub@off` commands appearing in the `\linenum@out` file.

```

1381 \renewcommand*{\@lab}{%
1382   \ifledRcol
1383     \xright@appenditem{\linenumr@p{\line@numR}}{|%
1384     \ifsublines@ \sublinenumr@p{\subline@numR}\else 0\fi}%
1385     \to\labelref@listR
1386   \else
1387     \xright@appenditem{\linenumr@p{\line@num}}{|%
1388     \ifsublines@ \sublinenumr@p{\subline@num}\else 0\fi}%
1389     \to\labelref@list
1390   \fi}
1391

```

17 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 1392 \newcount\sidenote@marginR
1393 \renewcommand*{\sidenotemargin}[1]{%
1394   \l@dgetsidenote@margin{#1}%
1395   \ifnum\@l@dttempcntb>\m@ne
1396     \ifledRcol
1397       \global\sidenote@marginR=\@l@dttempcntb
1398     \else
1399       \global\sidenote@margin=\@l@dttempcntb
1400     \fi
1401   \fi}}

```

```

1402 \sidenotemargin{right}
1403 \global\sidenote@margin=\@ne
1404

```

`\l@dlsnote` The ‘footnotes’ for left, right, and moveable sidenotes. The whole scheme is reminiscent of the critical footnotes code.

```

\l@dcsnote 1405 \renewcommand*{\l@dlsnote}[1]{%
1406   \begingroup%
1407   \newcommand{\content}{#1}%
1408   \ifnumberedpar@
1409     \ifledRcol%
1410       \xright@appenditem{\noexpand\l@dlsnote{\csexpandonce{content}}}%
1411                           \to\inserts@listR
1412       \global\advance\insert@countR \@ne%
1413     \else%
1414       \xright@appenditem{\noexpand\l@dlsnote{\csexpandonce{content}}}%
1415                           \to\inserts@list
1416       \global\advance\insert@count \@ne%
1417     \fi
1418   \fi\ignorespaces\endgroup}
1419 \renewcommand*{\l@drsnote}[1]{%
1420   \begingroup%
1421   \newcommand{\content}{#1}%
1422   \ifnumberedpar@
1423     \ifledRcol%
1424       \xright@appenditem{\noexpand\l@drsnote{\csexpandonce{content}}}%
1425                           \to\inserts@listR
1426       \global\advance\insert@countR \@ne%
1427     \else%
1428       \xright@appenditem{\noexpand\l@drsnote{\csexpandonce{content}}}%
1429                           \to\inserts@list
1430       \global\advance\insert@count \@ne%
1431     \fi
1432   \fi\ignorespaces\endgroup}
1433 \renewcommand*{\l@dcsnote}[1]{%
1434   \begingroup%
1435   \newcommand{\content}{#1}%
1436   \ifnumberedpar@
1437     \ifledRcol%
1438       \xright@appenditem{\noexpand\l@dcsnote{\csexpandonce{content}}}%
1439                           \to\inserts@listR
1440       \global\advance\insert@countR \@ne%
1441     \else%
1442       \xright@appenditem{\noexpand\l@dcsnote{\csexpandonce{content}}}%
1443                           \to\inserts@list
1444       \global\advance\insert@count \@ne%
1445     \fi
1446   \fi\ignorespaces\endgroup}
1447

```

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

```

1448 \newcommand*{\affixside@noteR}{%
1449   \def\sidenotecontent@{}%
1450   \numdef\itemcount@{0}%
1451   \def\do##1{%
1452     \ifnumequal{\itemcount@}{0}%
1453       {%
1454         \appto\sidenotecontent@{##1}}% Not print not separator before the 1st note
1455       {\appto\sidenotecontent@{\sidenotesep ##1}}%
1456       }%
1457     \numdef\itemcount@{\itemcount@+1}%
1458   }%
1459   \dolistloop{\l@dcstetext}%
1460   \ifnumgreater{\itemcount@}{1}{\eledmac@warning{\itemcount@\space sidenotes on line \the\line@numR
1461 \gdef\@templ@d{}}%
1462 \ifx\@templ@d\l@dcstetext \else%
1463   \if@twocolumn%
1464     \if@firstcolumn%
1465       \setl@dlp@rbox{\sidenotecontent@}%
1466     \else%
1467       \setl@drp@rbox{\sidenotecontent@}%
1468     \fi%
1469   \else%
1470     \l@dttempcntb=\sidenote@marginR%
1471     \ifnum\l@dttempcntb>\@ne%
1472       \advance\l@dttempcntb by\page@num%
1473     \fi%
1474     \ifodd\l@dttempcntb%
1475       \setl@drp@rbox{\sidenotecontent@t}%
1476     \else%
1477       \setl@dlp@rbox{\sidenotecontent@}%
1478     \fi%
1479   \fi%
1480 \fi}
1481
```

18 Familiar footnotes

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

```

1482 \renewcommand{\l@dbfnote}[1]{%
1483   \ifnumberedpar@
1484     \gdef\@tag{#1}%
1485     \ifledRcol%
1486       \xright@appenditem{\noexpand\vl@dbfnote{\csexpandonce{\@tag}}{\@thefnmark}}%
1487       \to\inserts@listR
1488       \global\advance\insert@countR \@ne%
1489     \else%

```

```

1490 \xright@appenditem{\noexpand\vl@dbfnote{\csexpandonce{@tag}}{\@thefnmark}}%
1491 \to\inserts@list
1492 \global\advance\insert@count \@ne%
1493 \fi
1494 \fi\ignorespaces}
1495
\normalbfnoteX
1496 \renewcommand{\normalbfnoteX}[2]{%
1497 \ifnumberedpar@
1498 \ifledRcol%
1499 \ifluatex
1500 \footnotelang@lua[R]%
1501 \fi
1502 \@ifundefined{xpg@main@language}%if polyglossia
1503 {}%
1504 {\footnotelang@poly[R]}%
1505 \protected@csxdef{thisfootnote}{\csuse{thefootnote#1}}%
1506 \xright@appenditem{\noexpand\vbfnoteX{#1}{#2}{\csexpandonce{thisfootnote}}}%
1507 \to\inserts@listR
1508 \global\advance\insert@countR \@ne%
1509 \else%
1510 \ifluatex
1511 \footnotelang@lua%
1512 \fi
1513 \@ifundefined{xpg@main@language}%if polyglossia
1514 {}%
1515 {\footnotelang@poly}%
1516 \protected@csxdef{thisfootnote}{\csuse{thefootnote#1}}%
1517 \xright@appenditem{\noexpand\vbfnoteX{#1}{#2}{\csexpandonce{thisfootnote}}}%
1518 \to\inserts@list
1519 \global\advance\insert@count \@ne%
1520 \fi
1521 \fi\ignorespaces}
1522

```

19 Verse

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

```

\inserthangingsymbolL
\inserthangingsymbolR 1523 \newif\ifinserthangingsymbolR
1524 \newcommand{\inserthangingsymbolL}{%
1525 \ifinserthangingsymbol%
1526 \ifinstanzaL%
1527 \hangingsymbol%
1528 \fi%
1529 \fi}

```

```

1530 \newcommand{\inserthangingsymbolR}{%
1531 \ifinserthangingsymbolR%
1532     \ifinstanzaR%
1533         \hangingsymbol%
1534     \fi%
1535 \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 1536 \newcommand{\correcthangingL}{%
1537 \ifl@dpaging\else%
1538     \ifinstanzaL%
1539         \ifinserthangingsymbol%
1540             \hskip \@ifundefined{sza@00}{0}{\expandafter%
1541                 \noexpand\csname sza@00\endcsname}\stanzaindentbase%
1542             \fi%
1543         \fi%
1544 \fi}
1545
1546 \newcommand{\correcthangingR}{%
1547 \ifl@dpaging\else%
1548     \ifinstanzaR%
1549         \ifinserthangingsymbolR%
1550             \hskip \@ifundefined{sza@00}{0}{\expandafter%
1551                 \noexpand\csname sza@00\endcsname}\stanzaindentbase%
1552             \fi%
1553         \fi%
1554 \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.

```

1555 \chardef\next=\catcode'\&
1556 \catcode'\&=\active
1557

```

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

```

1558 \newenvironment{astanza}{%
1559     \startstanzahook
1560     \catcode'\&=\active
1561     \global\stanza@count\@ne\stanza@modulo\@ne
1562     \ifnum\usernamecount{sza@00}=\z@
1563         \let\stanza@hang\relax
1564     \let\endlock\relax

```

```

1565 \else
1566 %% \interlinepenalty\@M % this screws things up, but I don't know why
1567 \rightskip\z@ plus 1fil\relax
1568 \fi
1569 \ifnum\usernamecount{szp00@}=\z@
1570 \let\sza@penalty\relax
1571 \fi
1572 \def&{%
1573 \endlock\mbox{}%
1574 \sza@penalty
1575 \global\advance\stanza@count\@ne
1576 \@astanza@line}%
1577 \def\&{%
1578 \endlock\mbox{}%
1579 \pend
1580 \endstanzaextra}%
1581 \pstart
1582 \@astanza@line
1583 }{}
1584

```

`\@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.

```

1585 \newcommand*{\@astanza@line}{%
1586 \ifnum\value{stanzaindentrepetition}=0
1587 \parindent=\csname sza@\number\stanza@count
1588 @\endcsname\stanzaindentbase
1589 \else
1590 \parindent=\csname sza@\number\stanza@modulo
1591 @\endcsname\stanzaindentbase
1592 \managestanza@modulo
1593 \fi
1594 \par
1595 \stanza@hang%\mbox{}%
1596 \ignorespaces}
1597

```

Lastly reset the modified category codes.

```

1598 \catcode'\&=\next
1599

```

20 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
\unvnamebox
\namebox

```

```

1600 \providecommand*\newnamebox}[1]{%
1601   \expandafter\newbox\csname #1\endcsname}
1602 \providecommand*\setnamebox}[1]{%
1603   \expandafter\setbox\csname #1\endcsname}
1604 \providecommand*\unhnamebox}[1]{%
1605   \expandafter\unhbox\csname #1\endcsname}
1606 \providecommand*\unvnamebox}[1]{%
1607   \expandafter\unvbox\csname #1\endcsname}
1608 \providecommand*\namebox}[1]{%
1609   \csname #1\endcsname}
1610

```

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

```

\usernamecount 1611 \providecommand*\newnamecount}[1]{%
1612   \expandafter\newcount\csname #1\endcsname}
1613 \providecommand*\usernamecount}[1]{%
1614   \csname #1\endcsname}
1615

```

21 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 default is 5120 chunk pairs.

```

1616 \newcount\l@dc@maxchunks
1617 \newcommand*\maxchunks}[1]{\l@dc@maxchunks=#1}
1618   \maxchunks{5120}
1619

```

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

```

\l@dnumpstartsR 1620 \newcount\l@dnumpstartsR
1621

```

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

```

\l@pscR 1622 \newcount\l@dpscL
1623 \newcount\l@dpscR
1624

```

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

```

1625 \newcommand*\l@dsetuprawboxes){%
1626   \@l@dttempcntb=\l@dc@maxchunks
1627   \loop\ifnum\@l@dttempcntb>\z@
1628     \newnamebox{\l@dLcolrawbox\the\@l@dttempcntb}

```

```

1629 \newnamebox{\l@Rcolrawbox\the\@l@tempcntb}
1630 \advance\@l@tempcntb \m@ne
1631 \repeat}
1632

```

`\l@dssetupmaxlinecounts` To be able to synchronise left and right texts we need to know the maximum number of text lines there are in each pair of chunks. `\l@dssetupmaxlinecounts` creates `\maxchunks` new counts called `\l@dsmaxlinesinpar1`, etc., and `\l@dszeromaxlinecounts` zeroes all of them.

```

1633 \newcommand*{\l@dssetupmaxlinecounts}{%
1634 \l@tempcntb=\l@dc@maxchunks
1635 \loop\ifnum\@l@tempcntb>\z@
1636 \newnamecount{\l@dsmaxlinesinpar\the\@l@tempcntb}
1637 \advance\@l@tempcntb \m@ne
1638 \repeat}
1639 \newcommand*{\l@dszeromaxlinecounts}{%
1640 \begingroup
1641 \l@tempcntb=\l@dc@maxchunks
1642 \loop\ifnum\@l@tempcntb>\z@
1643 \global\usenamecount{\l@dsmaxlinesinpar\the\@l@tempcntb}=\z@
1644 \advance\@l@tempcntb \m@ne
1645 \repeat
1646 \endgroup}
1647

```

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

```

1648 \AtBeginDocument{%
1649 \l@dssetuprawboxes
1650 \l@dssetupmaxlinecounts
1651 \l@dszeromaxlinecounts
1652 \l@dsnumpstartsL=\z@
1653 \l@dsnumpstartsR=\z@
1654 \l@dspscL=\z@
1655 \l@dspscR=\z@}
1656

```

22 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 1657 \newif\ifl@dusedbabel
\l@dusedbabeltrue 1658 \l@dusedbabelfalse

\ifl@dsamelang A flag for checking if the same babel language has been used for both the left and
\l@dsamelangfalse right texts.
\l@dsamelangtrue 1659 \newif\ifl@dsamelang
1660 \l@dsamelangtrue

\l@dchecklang I'm going to use \theledlanguageL and \theledlanguageR to hold the names of
the languages used for the left and right texts. This macro sets \ifl@dsamelang
TRUE if they are the same, otherwise it sets it FALSE.
1661 \newcommand*{\l@dchecklang}{%
1662 \l@dsamelangfalse
1663 \edef\@tempa{\theledlanguageL}\edef\@tempb{\theledlanguageR}%
1664 \ifx\@tempa\@tempb
1665 \l@dsamelangtrue
1666 \fi}
1667

\l@dbbl@set@language In babel the macro \bbl@set@language{<lang>} does the work when the language
<lang> is changed via \selectlanguage. Unfortunately for me, if it is given an
argument in the form of a control sequence it strips off the \ character rather than
expanding the command. I need a version that accepts an argument in the form
\lang without it stripping the \.
1668 \newcommand*{\l@dbbl@set@language}[1]{%
1669 \edef\languageName{#1}%
1670 \select@language{\languageName}%
1671 \if@filesw
1672 \protected@write\auxout{}\string\select@language{\languageName}}%
1673 \addtocontents{toc}{\string\select@language{\languageName}}%
1674 \addtocontents{lof}{\string\select@language{\languageName}}%
1675 \addtocontents{lot}{\string\select@language{\languageName}}%
1676 \fi}
1677

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 1678 \providecommand{\selectlanguage}[1]{%
1679 \newcommand*{\l@duselanguage}[1]{%
1680 \gdef\theledlanguageL{
1681 \gdef\theledlanguageR{
1682

```

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

```
1683 \AtBeginDocument{%
1684   \ifundefined{xpg@main@language}{%
1685     \ifundefined{bbl@main@language}{%
```

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

```
1686   \l@dusedbabelfalse
1687   \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.

```
1688   \l@dusedbabeltrue
1689   \let\l@doldselectlanguage\selectlanguage
1690   \let\l@doldbbl@set@language\bbl@set@language
1691   \let\bbl@set@language\l@doldbbl@set@language
1692   \renewcommand{\selectlanguage}[1]{%
1693     \l@doldselectlanguage{#1}%
1694     \ifledRcol \gdef\theledlanguageR{#1}%
1695     \else      \gdef\theledlanguageL{#1}%
1696     \fi}
```

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

```
1697   \renewcommand*{\l@duselanguage}[1]{%
1698     \l@doldselectlanguage{#1}}
```

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

```
1699   \gdef\theledlanguageL{\bbl@main@language}%
1700   \gdef\theledlanguageR{\bbl@main@language}%
1701   }%
1702 }
```

If on `Polyglossia`

```
1703 { \apptocmd{\xpg@set@language}{%
1704   \ifledRcol \gdef\theledlanguageR{#1}%
1705   \else      \gdef\theledlanguageL{#1}%
1706   \fi}%
1707   \let\l@duselanguage\xpg@set@language
1708   \gdef\theledlanguageL{\xpg@main@language}%
1709   \gdef\theledlanguageR{\xpg@main@language}%
1710 % \end{macrocode}
1711 % That's it.
1712 %   \begin{macrocode}
1713 }}
```

23 Parallel columns

`\Columns` The `\Columns` command results in the previous Left and Right texts being typeset in matching columns. There should be equal numbers of chunks in the left and

right texts.

```

1714 \newcommand*{\Columns}{%
1715   \setcounter{pstartL}{\value{pstartLold}}
1716   \setcounter{pstartR}{\value{pstartRold}}
1717   \ifnum\l@dnumstartsL=\l@dnumstartsR\else
1718     \led@err@BadLeftRightPstarts{\the\l@dnumstartsL}{\the\l@dnumstartsR}%
1719   \fi

```

Start a group and zero counters, etc.

```

1720 \begingroup
1721   \l@dzeropenalties
1722   \endgraf\global\num@lines=\prevgraf
1723   \global\num@linesR=\prevgraf
1724   \global\par@line=\z@
1725   \global\par@lineR=\z@
1726   \global\l@dpscL=\z@
1727   \global\l@dpscR=\z@

```

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

```

1728   \check@pstarts
1729   \loop\if@pstarts
1730     \global\pstartnumtrue
1731     \global\pstartnumRtrue

```

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

```

1732   \global\advance\l@dpscL \@ne
1733   \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

```

1734   \checkraw@text
1735   \l@dchecklang
1736 {   \loop\ifaraw@text

```

Grab the next pair of left and right text lines and output them, swapping languages if they differ

```

1737     \ifl@dsamelang
1738       \do@lineL
1739       \do@lineR
1740     \else
1741       \l@duselanguage{\theledlanguageL}%
1742       \do@lineL
1743       \l@duselanguage{\theledlanguageR}%
1744       \do@lineR
1745     \fi
1746     \hb@xt@ \hsize{%
1747       \hfill \unhbox\l@dleftbox
1748       \hfill \columnseparator \hfill
1749       \unhbox\l@drightbox

```

```

1750      }%
1751      \checkraw@text
1752      \checkverseL
1753      \checkverseR
1754      \checkpb@columns
1755      \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.

```

1756      \@writelinesinparL
1757      \@writelinesinparR
1758      \check@pstarts
1759      \ifbypstart@
1760      \write\linenum@out{\string\@set[1]}
1761      \resetprevline@
1762      \fi
1763      \ifbypstart@R
1764      \write\linenum@outR{\string\@set[1]}
1765      \resetprevline@
1766      \fi
1767      \addtocounter{pstartL}{1}
1768      \addtocounter{pstartR}{1}
1769      \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.

```

1770      \flush@notes
1771      \flush@notesR
1772      \endgroup
1773      \global\l@dpscL=\z@
1774      \global\l@dpscR=\z@
1775      \global\l@dnumpstartsL=\z@
1776      \global\l@dnumpstartsR=\z@
1777      \ignorespaces
1778      \global\instanzaLfalse
1779      \global\instanzaRfalse}
1780

```

`\checkpb@columns` `\checkpb@columns` prevent or make pagebreaking in columns, depending of the use of `\ledpb` or `\lednopb`.

```

1781
1782 \newcommand{\checkpb@columns}{%
1783   \newif\if@pb
1784   \newif\if@nopb
1785   \IfStrEq{\led@pb@setting}{before}{
1786     \numdef{\next@absline}{\the\absline@num+1}%
1787     \numdef{\next@abslineR}{\the\absline@numR+1}%
1788     \xifinlistcs{\next@absline}{l@prev@pb}{\@pbtrue}{}%
1789     \xifinlistcs{\next@abslineR}{l@prev@pbR}{\@pbtrue}{%

```

```

1790 \xifinlistcs{\next@absline}{l@prev@nopb}{\@nopbtrue}{}%
1791 \xifinlistcs{\next@abslineR}{l@prev@nopbR}{\@nopbtrue}{}
1792 }{}
1793 \IfStrEq{\led@pb@setting}{after}{
1794 \xifinlistcs{\the\absline@num}{l@prev@pb}{\@pbtrue}{}%
1795 \xifinlistcs{\the\absline@numR}{l@prev@pbR}{\@pbtrue}{}
1796 \xifinlistcs{\the\absline@num}{l@prev@nopb}{\@nopbtrue}{}%
1797 \xifinlistcs{\the\absline@numR}{l@prev@nopbR}{\@nopbtrue}{}
1798 }{}
1799 \if@nopb\nopagebreak[4]\enlargethispage{\baselineskip}\fi
1800 \if@pb\pagebreak[4]\fi
1801 }

```

`\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).

```

1802 \newcommand*{\columnseparator}{%
1803 \smash{\rule[-0.2\baselineskip]{\columnrulewidth}{1.05\baselineskip}}}
1804 \newdimen\columnrulewidth
1805 \columnrulewidth=\z@
1806

```

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

```

\@pstartstrue 1807 \newif\if@pstarts
\@pstartstrue 1808 \newcommand*{\check@pstarts}{%
\check@pstarts 1809 \ifnum\l@dnumpstartsL>\l@dpscl
1810 \ifnum\l@dnumpstartsL>\l@dpscl
1811 \@pstartstrue
1812 \else
1813 \ifnum\l@dnumpstartsR>\l@dpscl
1814 \@pstartstrue
1815 \fi
1816 \fi
1817 }
1818

```

`\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 1819 \newif\ifaraw@text
1820 \araw@textfalse
1821 \newcommand*{\checkraw@text}{%
1822 \araw@textfalse
1823 \ifvbox\namebox{l@dLcolrawbox\the\l@dpscl}
1824 \araw@texttrue
1825 \else
1826 \ifvbox\namebox{l@dRcolrawbox\the\l@dpscl}
1827 \araw@texttrue

```

```

1828     \fi
1829   \fi
1830 }
1831

```

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

```

1832 \newcommand*{\@writelinesinparL}{%
1833   \edef\next{%
1834     \write\linenum@out{\string\@pend[\the\@donereallinesL]}}%
1835   \next
1836   \global\@donereallinesL \z@}
1837 \newcommand*{\@writelinesinparR}{%
1838   \edef\next{%
1839     \write\linenum@outR{\string\@pendR[\the\@donereallinesR]}}%
1840   \next
1841   \global\@donereallinesR \z@}
1842

```

24 Parallel pages

This is considerably more complicated than parallel columns.

`\numpagelinesL` Counts for the number of lines on a left or right page, and the smaller of the
`\numpagelinesR` number of lines on a pair of facing pages.
`\l@dminpagelines` 1843 \newcount\numpagelinesL
1844 \newcount\numpagelinesR
1845 \newcount\l@dminpagelines
1846

`\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.

```

1847 \newcommand*{\Pages}{%
1848   \setcounter{pstartL}{\value{pstartLold}}
1849   \setcounter{pstartR}{\value{pstartRold}}
1850   \typeout{}
1851   \typeout{***** PAGES *****}
1852   \ifnum\l@dnumstartsL=\l@dnumstartsR\else
1853     \led@err@BadLeftRightPstarts{\the\l@dnumstartsL}{\the\l@dnumstartsR}%
1854   \fi

```

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

```

1855 \cleartol@devenpage
1856 \begingroup
1857   \l@dzeropenalties
1858   \endgraf\global\num@lines=\prevgraf
1859   \global\num@linesR=\prevgraf

```

```

1860 \global\par@line=\z@
1861 \global\par@lineR=\z@
1862 \global\l@dpscL=\z@
1863 \global\l@dpscR=\z@
1864 \writtenlinesLfalse
1865 \writtenlinesRfalse

```

Check if there are chunks to be processed.

```

1866 \check@pstarts
1867 \loop\if@pstarts

```

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

```

1868 \global\advance\l@dpscL \@ne
1869 \global\advance\l@dpscR \@ne

```

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

```

1870 \getlinesfromparlistL
1871 \getlinesfromparlistR
1872 \l@dcalc@maxoftwo{\@cs@linesinparL}{\@cs@linesinparR}%
1873 {\usernamecount{\l@dmaxlinesinpar\the\l@dpscL}}%
1874 \check@pstarts
1875 \repeat

```

Zero the counts again, ready for the next bit.

```

1876 \global\l@dpscL=\z@
1877 \global\l@dpscR=\z@

```

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

```

1878 \getlinesfrompagelistL
1879 \getlinesfrompagelistR
1880 \l@dcalc@minoftwo{\@cs@linesonpageL}{\@cs@linesonpageR}%
1881 {\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.

```

1882 \check@pstarts
1883 \if@pstarts

```

Increment the chunk counts to get the first pair.

```

1884 \global\advance\l@dpscL \@ne
1885 \global\advance\l@dpscR \@ne

```

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

```

1886 \global\@donereallinesL=\z@
1887 \global\@donetotallinesL=\z@
1888 \global\@donereallinesR=\z@
1889 \global\@donetotallinesR=\z@

```

Start a loop over the boxes (chunks).

```
1890      \checkraw@text
1891 %      \begingroup
1892 {      \loop\ifaraw@text
```

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

```
1893      \checkpageL
1894      \l@duselanguage{\theledlanguageL}%
1895 %%%      \begingroup
1896 {      \loop\ifl@dsamepage
1897
```

Process the next (left) text line, adding it to the page.

```
1898      \do@lineL
1899      \advance\numpagelinesL \@ne
1900      \ifshiftedpstarts
1901          \ifdim\ht\l@dleftbox>0pt\hb@xt@ \hsize{\ledstrutL\unhbox\l@dleftbox}%
1902      \else%
1903          \hb@xt@ \hsize{\ledstrutL\unhbox\l@dleftbox}%
1904      \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.

```
1905
1906      \get@nextboxL
1907      \checkpageL
1908      \checkverseL
1909      \checkpbL
1910      \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.

```
1911      \ifl@dpagfull
1912      \@writelinesonpageL{\the\numpagelinesL}%
1913      \else
1914      \@writelinesonpageL{1000}%
1915      \fi
```

Zero the left page lines count and clear the page to get onto the facing (odd, right) page.

```
1916      \numpagelinesL \z@
1917      \clearl@dleftpage }%
```

Now do the same for the right text.

```
1918      \checkpageR
1919      \l@duselanguage{\theledlanguageR}%
1920 {      \loop\ifl@dsamepage
1921      \do@lineR
```

```

1922         \advance\numpagelinesR \@ne
1923         \ifshiftedpstarts
1924             \ifdim\ht\l@drightbox>0pt\hb@xt@ \hsize{\ledstrutR\unhbox\l@drightbox}\fi%
1925         \else%
1926             \hb@xt@ \hsize{\ledstrutR\unhbox\l@drightbox}%
1927         \fi
1928         \get@nextboxR
1929         \checkpageR
1930         \checkverseR
1931         \checkpbR
1932         \repeat
1933         \ifl@dpagfull
1934             \@writelinesonpageR{\the\numpagelinesR}%
1935         \else
1936             \@writelinesonpageR{1000}%
1937         \fi
1938         \numpagelinesR=\z@

```

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

```

1939         \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.

```

1940         \checkraw@text
1941         \ifaraw@text
1942             \getlinesfrompagelistL
1943             \getlinesfrompagelistR
1944             \l@dcalc@minoftwo{\@cs@linesonpageL}{\@cs@linesonpageR}%
1945                                 {\l@dminpagelines}%
1946         \fi
1947         \repeat}

```

We have now output the text from all the chunks.

```

1948         \fi

```

Make sure that there are no inserts hanging around.

```

1949         \flush@notes
1950         \flush@notesR
1951     \endgroup

```

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

```

1952     \global\l@dpscL=\z@
1953     \global\l@dpscR=\z@
1954     \global\l@dnpstartsL=\z@
1955     \global\l@dnpstartsR=\z@
1956     \global\instanzaLfalse
1957     \global\instanzaRfalse
1958     \ignorespaces}
1959

```

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

`\ledstrutR` 1960 `\newcommand*{\ledstrutL}{\strut}`
1961 `\newcommand*{\ledstrutR}{\strut}`
1962

`\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.

1963 `\providecommand{\cleartoevenpage}[1][\@empty]{%`
1964 `\clearpage`
1965 `\ifodd\c@page\hbox{#1\clearpage}\fi}`
1966 `\newcommand*{\cleartol@devenpage}{%`
1967 `\ifdim\pagetotal<\topskip% on an empty page`
1968 `\else`
1969 `\clearpage`
1970 `\fi`
1971 `\ifodd\c@page\hbox{ }\clearpage}\fi}`
1972 `\newcommand*{\clearl@dleftpage}{%`
1973 `\clearpage`
1974 `\ifodd\c@page\else`
1975 `\led@err@LeftOnRightPage`
1976 `\hbox{ }%`
1977 `\cleardoublepage`
1978 `\fi}`
1979 `\newcommand*{\clearl@drightpage}{%`
1980 `\clearpage`
1981 `\ifodd\c@page`
1982 `\led@err@RightOnLeftPage`
1983 `\hbox{ }%`
1984 `\cleartoevenpage`
1985 `\fi}`
1986

`\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` 1987 `\newcommand*{\getlinesfromparlistL}{%`
1988 `\ifx\linesinpar@listL\empty`
1989 `\gdef\@cs@linesinparL{0}%`
1990 `\else`
1991 `\gl@p\linesinpar@listL\to\@cs@linesinparL`
1992 `\fi}`
1993 `\newcommand*{\getlinesfromparlistR}{%`
1994 `\ifx\linesinpar@listR\empty`
1995 `\gdef\@cs@linesinparR{0}%`
1996 `\else`
1997 `\gl@p\linesinpar@listR\to\@cs@linesinparR`

1998 \fi}
 1999

\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 2000 \newcommand*\getlinesfrompagelistL{%
2001   \ifx\linesonpage@listL\empty
2002     \gdef\cs@linesonpageL{1000}%
2003   \else
2004     \gl@p\linesonpage@listL\to\cs@linesonpageL
2005   \fi}
2006 \newcommand*\getlinesfrompagelistR{%
2007   \ifx\linesonpage@listR\empty
2008     \gdef\cs@linesonpageR{1000}%
2009   \else
2010     \gl@p\linesonpage@listR\to\cs@linesonpageR
2011   \fi}
2012
```

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

```
2013 \newcommand*\@writelinesonpageL[1]{%
2014   \edef\next{\write\linenum@out{\string\@lopL{#1}}}%
2015   \next}
2016 \newcommand*\@writelinesonpageR[1]{%
2017   \edef\next{\write\linenum@outR{\string\@lopR{#1}}}%
2018   \next}
2019
```

\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>.

```
2020 \newcommand*\l@dcalc@maxoftwo[3]{%
2021   \ifnum #2>#1\relax
2022     #3=#2\relax
2023   \else
2024     #3=#1\relax
2025   \fi}
2026 \newcommand*\l@dcalc@minoftwo[3]{%
2027   \ifnum #2<#1\relax
2028     #3=#2\relax
2029   \else
2030     #3=#1\relax
2031   \fi}
2032
```

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

```

\checkpageL 2033 \newif\ifl@dsamepage
\checkpageR 2034 \l@dsamepagetrue
2035 \newif\ifl@dpagfull
2036
2037 \newcommand*\checkpageL}{%
2038   \l@dpagfulltrue
2039   \l@dsamepagetrue
2040   \check@goal
2041   \ifdim\pagetotal<\ledthegoal
2042     \ifnum\numpagelinesL<\l@dminpagelines
2043       \else
2044         \l@dsamepagefalse
2045         \l@dpagfullfalse
2046       \fi
2047     \else
2048       \l@dsamepagefalse
2049       \l@dpagfulltrue
2050     \fi}
2051 \newcommand*\checkpageR}{%
2052   \l@dpagfulltrue
2053   \l@dsamepagetrue
2054   \check@goal
2055   \ifdim\pagetotal<\ledthegoal
2056     \ifnum\numpagelinesR<\l@dminpagelines
2057       \else
2058         \l@dsamepagefalse
2059         \l@dpagfullfalse
2060       \fi
2061     \else
2062       \l@dsamepagefalse
2063       \l@dpagfulltrue
2064     \fi}
2065

```

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

```

2066 \newcommand*\checkpbL){
2067   \IfStrEq{\led@pb@setting}{after}{
2068     \xifinlistcs{\the\absline@num}{\l@prev@pb}{\l@dpagfulltrue\l@dsamepagefalse}{
2069       \xifinlistcs{\the\absline@num}{\l@prev@nopb}{\l@dpagfullfalse\l@dsamepagetrue}{
2070       }{
2071     \IfStrEq{\led@pb@setting}{before}{

```

```

2072 \numdef{\next@absline}{\the\absline@num+1}
2073 \xifinlistcs{\next@absline}{\l@prev@pb}{\l@dpagfulltrue\l@dsamepagefalse}{\l@dpagfullfalse\l@dsamepagetrue}{\l@dpagfulltrue\l@dsamepagefalse}{\l@dpagfullfalse\l@dsamepagetrue}}
2074 \xifinlistcs{\next@absline}{\l@prev@nopb}{\l@dpagfullfalse\l@dsamepagetrue}{\l@dpagfulltrue\l@dsamepagefalse}{\l@dpagfullfalse\l@dsamepagetrue}}
2075 }{}
2076 }
2077
2078 \newcommand{\checkpbR}{
2079 \IfStrEq{\led@pb@setting}{after}{
2080 \xifinlistcs{\the\absline@numR}{\l@prev@pbR}{\l@dpagfulltrue\l@dsamepagefalse}{\l@dpagfullfalse\l@dsamepagetrue}}
2081 \xifinlistcs{\the\absline@numR}{\l@prev@nopbR}{\l@dpagfullfalse\l@dsamepagetrue}{\l@dpagfulltrue\l@dsamepagefalse}{\l@dpagfullfalse\l@dsamepagetrue}}
2082 }{}
2083 \IfStrEq{\led@pb@setting}{before}{
2084 \numdef{\next@abslineR}{\the\absline@numR+1}
2085 \xifinlistcs{\next@abslineR}{\l@prev@pbR}{\l@dpagfulltrue\l@dsamepagefalse}{\l@dpagfullfalse\l@dsamepagetrue}}
2086 \xifinlistcs{\next@abslineR}{\l@prev@nopbR}{\l@dpagfullfalse\l@dsamepagetrue}{\l@dpagfulltrue\l@dsamepagefalse}{\l@dpagfullfalse\l@dsamepagetrue}}
2087 }{}
2088 }

```

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

```

2089 \newcommand{\checkverseL}{%
2090 \ifinstanzaL
2091   \iflednopbinverse
2092     \ifinserthangingsymbol
2093       \numgdef{\prev@abslineverse}{\the\absline@num-1}
2094       \IfStrEq{\led@pb@setting}{after}{\lednopbnum{\prev@abslineverse}}{}
2095       \IfStrEq{\led@pb@setting}{before}{\ifnum\numpagelinesL<3\ledpbnum{\prev@abslineverse}\fi}{%
2096         \fi
2097       \fi
2098 \fi
2099 }
2100 \newcommand{\checkverseR}{%
2101 \ifinstanzaR
2102   \iflednopbinverse
2103     \ifinserthangingsymbolR
2104       \numgdef{\prev@abslineverse}{\the\absline@numR-1}
2105       \IfStrEq{\led@pb@setting}{after}{\lednopbnumR{\prev@abslineverse}}{}
2106       \IfStrEq{\led@pb@setting}{before}{\ifnum\numpagelinesR<3\ledpbnumR{\prev@abslineverse}\fi}{%
2107         \fi
2108       \fi
2109 \fi
2110 }

```

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

```

2111 \newdimen\ledthegoal
2112 \ifshiftedpstarts
2113     \newcommand*{\goalfraction}{0.95}

```

```

2114 \else
2115     \newcommand*{\goalfraction}{0.9}
2116 \fi
2117
2118 \newcommand*{\check@goal}{%
2119     \ledthegoal=\goalfraction\pagegoal}
2120
\ifwrittenlinesL    Booleans for whether line data has been written to the section file.
\ifwrittenlinesL 2121 \newif\ifwrittenlinesL
2122 \newif\ifwrittenlinesR
2123

\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.

2124 \newcommand*{\get@nextboxL}{%
2125     \ifvbox\namebox{\l@dLcolrawbox\the\l@dpscL}% box is not empty

        The current box is not empty; do nothing.
2126 \else%                                box is empty

        The box is empty; check if enough lines (real and blank) have been output.
2127     \ifnum\usernamecount{\l@dmxlinesinpar\the\l@dpscL}>\@donetotallinesL
2128     \else

        Sufficient lines have been output.
2129         \ifwrittenlinesL
2130         \else

        Write out the number of lines done, and set the boolean so this is only done once.
2131             \@writelinesinparL
2132             \writtenlinesLtrue
2133         \fi
2134         \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.
2135             \writtenlinesLfalse
2136             \ifbypstart@
2137                 \ifnum\value{pstartL}<\value{pstartLold}
2138                 \else
2139                     \global\line@num=0
2140                     \resetprevline@
2141                 \fi
2142             \fi
2143             \addtocounter{pstartL}{1}
2144             \global\pstartnumtrue
2145             \l@dcalc@maxoftwo{\the\usernamecount{\l@dmxlinesinpar\the\l@dpscL}}%
2146                             {\the\@donetotallinesL}%
2147                             {\usernamecount{\l@dmxlinesinpar\the\l@dpscL}}%

```

```

2148      \global\@donetotallinesL \z@
2149      \global\advance\l@dpscL \@ne
2150      \fi
2151      \fi
2152      \fi}

2153 \newcommand*\get@nextboxR{%
2154   \ifvbox\namebox{\l@dRcolrawbox\the\l@dpscR}% box is not empty
2155   \else% box is empty
2156     \ifnum\usernamecount{\l@dmaxlinesinpar\the\l@dpscR}>\@donetotallinesR
2157     \else
2158       \ifwrittenlinesR
2159       \else
2160         \@writelinesinparR
2161         \writtenlinesRtrue
2162       \fi
2163       \ifnum\l@dnumstartsR>\l@dpscR
2164         \writtenlinesRfalse
2165         \ifbypstart@R
2166           \ifnum\value{pstartR}<\value{pstartRold}
2167           \else
2168             \global\line@numR=0
2169             \resetprevline@
2170           \fi
2171         \fi
2172         \addtocounter{pstartR}{1}
2173         \global\pstartnumRtrue
2174         \l@dcalcl@maxoftwo{\the\usernamecount{\l@dmaxlinesinpar\the\l@dpscR}}%
2175           {\the\@donetotallinesR}%
2176         {\usernamecount{\l@dmaxlinesinpar\the\l@dpscR}}}%
2177         \global\@donetotallinesR \z@
2178         \global\advance\l@dpscR \@ne
2179       \fi
2180     \fi
2181   \fi}
2182

```

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

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

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

```

2183 \def\l@prev@pbR{}
2184 \def\l@prev@nopbR{}

```

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

```
2185 \newcommand{\ledpbR}{\write\linenum@outR{\string\led@pbR}}
2186 \newcommand{\ledpbnumR}[1]{\write\linenum@outR{\string\led@pbnumR{#1}}}
2187 \newcommand{\lednopbR}{\write\linenum@outR{\string\led@nopbR}}
2188 \newcommand{\lednopbnumR}[1]{\write\linenum@outR{\string\led@nopbnumR{#1}}}
```

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

```
2189 \newcommand{\led@pbR}{\listcsxadd{1@prev@pbR}{\the\absline@numR}}
2190 \newcommand{\led@pbnumR}[1]{\listcsxadd{1@prev@pbR}{#1}}
2191 \newcommand{\led@nopbR}{\listcsxadd{1@prev@nopbR}{\the\absline@numR}}
2192 \newcommand{\led@nopbnumR}[1]{\listcsxadd{1@prev@nopbR}{#1}}
```

26 The End

`i/codei`

Appendix A Some things to do when changing version

Appendix A.1 Migration to eledpar 1.4.3

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

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

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

See the two following examples:

With standard `\hangingsymbol`:

A very long verse should be sometime hanged. The position of the hang 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>\@M</code>	1566
<code>\&</code>	1555, 1556, 1560, 1577, 1598	<code>\@adv</code> <u>359</u> , 640, 641

- \@afterindentfalse 763
 - \@arabic 215, 216, 814, 817
 - \@astanza@line 1576, 1582, 1585
 - \@auxout 1354, 1366, 1672
 - \@chapter 764
 - \@cs@linesinparL 1872, 1987
 - \@cs@linesinparR 1872, 1987
 - \@cs@linesonpageL .. 1880, 1944, 2000
 - \@cs@linesonpageR .. 1880, 1944, 2000
 - \@currentlabel 857, 897
 - \@donereallinesL
 - 941, 970, 1834, 1836, 1886
 - \@donereallinesR
 - 941, 1007, 1839, 1841, 1888
 - \@donetotallinesL 941,
 - 971, 974, 1887, 2127, 2146, 2148
 - \@donetotallinesR 941,
 - 1008, 1011, 1889, 2156, 2175, 2177
 - \@insertR 1308–1310, 1323–1325
 - \@l 281, 607, 609
 - \@l@dttempcnta 432,
 - 434, 436, 437, 441, 443, 445,
 - 446, 1061, 1100, 1101, 1103,
 - 1105, 1108, 1109, 1126–1130,
 - 1132, 1139, 1144, 1148, 1156,
 - 1161, 1165, 1198, 1201, 1203, 1207
 - \@l@dttempcntb . 171, 173, 175, 1091,
 - 1092, 1139, 1144, 1148, 1156,
 - 1161, 1165, 1190, 1194, 1207,
 - 1215–1217, 1219, 1239–1241,
 - 1243, 1260–1262, 1264, 1395,
 - 1397, 1399, 1470–1472, 1474,
 - 1626–1630, 1634–1637, 1641–1644
 - \@l@reg 330
 - \@l@regR 281
 - \@lab 551, 1345, 1357, 1381
 - \@lock 957, 1042
 - \@lockR 63, 303, 305, 307, 320,
 - 466, 482, 483, 485, 486, 514,
 - 515, 517, 994, 1025, 1067, 1069,
 - 1070, 1072, 1153, 1170, 1172, 1174
 - \@lopL 575, 2014
 - \@lopR 575, 2017
 - \@nobreakfalse 823, 864
 - \@nobreaktrue 821, 825, 862, 866
 - \@nobbtrue 1790, 1791, 1796, 1797
 - \@oldnobreak 821, 823, 862, 864, 911, 927
 - \@pbtrue 1788, 1789, 1794, 1795
 - \@pend 566, 1834
 - \@pendR 566, 1839
 - \@pstartsfalse 1807
 - \@pstartstrue 1807
 - \@ref 538, 613, 617
 - \@ref@reg 564
 - \@schapter 764
 - \@set 391, 647, 648, 1760, 1764
 - \@tag 677, 700, 1484
 - \@temp 1663
 - \@templ@ 1461, 1462
 - \@writelinesinparL .. 1756, 1832, 2131
 - \@writelinesinparR .. 1757, 1832, 2160
 - \@writelinesonpageL . 1912, 1914, 2013
 - \@writelinesonpageR . 1934, 1936, 2013
 - \@xloop 1321
- A**
- \absline@num
 - 355, 425, 439, 458, 1033, 1786,
 - 1794, 1796, 2068, 2069, 2072, 2093
 - \absline@numR 58, 232, 283,
 - 286, 289, 422, 430, 451, 470,
 - 504, 532, 543, 1016, 1053, 1054,
 - 1091, 1307, 1787, 1795, 1797,
 - 2080, 2081, 2084, 2104, 2189, 2191
 - \actionlines@list
 - 273, 276, 425, 439, 458
 - \actionlines@listR
 - 236, 251, 265, 268, 422,
 - 430, 451, 470, 504, 532, 1113, 1116
 - \actions@list . 277, 426, 446, 460, 462
 - \actions@listR
 - . 236, 252, 269, 423, 437, 453,
 - 455, 472, 481, 506, 513, 533, 1117
 - \add@inserts 963
 - \add@inserts@nextR 1296
 - \add@insertsR 1000, 1296
 - \add@penaltiesL 969, 1317
 - \add@penaltiesR 1006, 1317
 - \addtocontents 1673–1675
 - \addtocounter
 - . 913, 929, 1767, 1768, 2143, 2172
 - \advancelabel@refs 1352, 1364
 - \advanceline 639, 670
 - \affixline@num 961
 - \affixline@numR 998, 1123
 - \affixpstart@numL 960, 1229
 - \affixpstart@numR 997, 1229
 - \affixside@note 964
 - \affixside@noteR 1001, 1448
 - \appto 1454, 1455

\apptocmd 1703
 \araw@textfalse 1819
 \araw@texttrue 1819
 astanza (environment) 9, 1558
 \AtBeginDocument ... 1377, 1648, 1683

B

\ballast@count 1051, 1056
 \bbl@main@language 1699, 1700
 \bbl@set@language 1690, 1691
 \beginnumbering .. 8, 36, 770, 788, 828
 \beginnumberingR ... 49, 120, 788, 869
 \bfseries 814, 817
 \bypage@Rfalse 140, 155, 160
 \bypage@Rtrue 140, 150
 \bypstart@Rfalse 140, 151, 161
 \bypstart@Rtrue 140, 156

C

\c@ballast 1056
 \c@chapter 103
 \c@chapterR 103
 \c@firstlinenumR 180, 1196
 \c@firstsublinenumR 184, 1191
 \c@linenumincrementR 180, 1196
 \c@page 607, 609, 1965, 1971, 1974, 1981
 \c@pstartL 814
 \c@pstartR 817
 \c@section 104
 \c@sectionR 104
 \c@sublinenumincrementR .. 184, 1191
 \c@subsection 105
 \c@subsectionR 105
 \c@subsubsection 106
 \c@subsubsectionR 106
 \ch@ck@l@ckR 1123
 \ch@cksub@l@ckR 1123
 \ch@cksub@lockR 1192
 \chapter 750, 751, 759
 \chapterinpages 743, 751, 761
 \chardef 1555
 \check@goal 2040, 2054, 2111
 \check@pstarts
 1728, 1758, 1807, 1866, 1874, 1882
 \checkpageL 1893, 1907, 2033
 \checkpageR 1918, 1929, 2033
 \checkpb@columns 1754, 1781
 \checkpbL 1909, 2066
 \checkpbR 1931, 2066

\checkraw@text
 1734, 1751, 1819, 1890, 1940
 \checkverseL 1752, 1908, 2089
 \checkverseR 1753, 1930, 2089
 \cleardoublepage 1977
 \clearl@dleftpage 1917, 1963
 \clearl@drihtpage 1939, 1963
 \cleartoevenpage 1963
 \cleartol@devenpage 1855, 1963
 \closeout 588, 594, 597, 601
 \columnrulewidth 5, 1802
 \Columns 5, 1714
 \columnseparator 5, 1748, 1802
 \content 1407, 1421, 1435
 \correcthangingL 966, 1536
 \correcthangingR 1003, 1536
 \countLline 936, 947
 \countRline 936, 984
 \critext 675
 \csexpandonce 1410, 1414, 1424, 1428,
 1438, 1442, 1486, 1490, 1506, 1517
 \csuse 1505, 1516

D

\DeclareOption 8, 9
 \def@tempb 158
 \dimen 626, 627, 631–633, 637
 \divide 1128
 \do@actions 1034
 \do@actions@fixedcodeR 1060
 \do@actions@nextR 1060
 \do@actionsR 1017, 1060
 \do@ballast 1035
 \do@ballastR 1018, 1051
 \do@insidelineLhook 966, 978
 \do@insidelineRhook 978, 1003
 \do@lineL 946, 1738, 1742, 1898
 \do@lineLhook 951, 978
 \do@lineR 983, 1739, 1744, 1921
 \do@lineRhook 978, 988
 \do@lockoff 501
 \do@lockoffL 525
 \do@lockoffR 501
 \do@lockon 466
 \do@lockonL 498
 \do@lockonR 466
 \dolistloop 1459
 \dummy@ref 547

E

`\edfont@info` 726, 729, 735, 738
`\edlabel` 1343
`\edtext` 698
`\eledmac@error` 21, 24, 27, 29
`\eledmac@warning` 1460
`\empty` .. 84, 87, 265, 273, 688, 711,
 724, 733, 837, 878, 1113, 1195,
 1203, 1298–1300, 1311, 1322,
 1346, 1358, 1988, 1994, 2001, 2007
`\end@lemmas` 688, 689, 711, 712
`\endashchar` 1333
`\endgraf` 907, 923, 1722, 1858
`\endline@num` 554, 560
`\endlock` 659, 1564, 1573, 1578
`\endnumbering` 8, 39, 77, 124, 789
`\endnumberingR` 52, 77, 109, 119, 132, 789
`\endpage@num` 553, 560
`\endstanzaextra` 1580
`\endsub` 626
`\endsubline@num` 555, 561
`\enlargethispage` 1799
environments:
 `astanza` 9, 1558
 `Leftside` 6, 768
 `pages` 5, 743
 `pairs` 5, 743
 `Rightside` 6, 786
`\extensionchars` . 47, 69, 115, 129, 137

F

`\f@x@l@cksR` 1123
`\first@linenum@out@Rfalse` .. 583, 589
`\first@linenum@out@Rtrue` 583
`\firstlinenum` 6, 189
`\firstlinenum*` 6, 189
`\firstsublinenum` 6, 189
`\firstsublinenum*` 6, 189
`\fix@page` 326, 333
`\flag@end`
 . 610, 684, 694, 695, 707, 717, 718
`\flag@start`
 . 610, 683, 684, 695, 706, 707, 718
`\flush@notes` 1770, 1949
`\flush@notesR` 1320, 1771, 1950
`\footnotelang@lua` 1500, 1511
`\footnotelang@poly` 1504, 1515
`\fullstop` . 228, 1330, 1332, 1334, 1336

G

`\get@linelistfile` 261
`\get@nextboxL` 1906, 2124
`\get@nextboxR` 1928, 2124
`\getline@numL` 956, 1032
`\getline@numR` 993, 1015
`\getlinesfrompagelistL`
 1878, 1942, 2000
`\getlinesfrompagelistR`
 1879, 1943, 2000
`\getlinesfromparlistL` ... 1870, 1987
`\getlinesfromparlistR` ... 1871, 1987
`\gl@p` 268, 269,
 276, 277, 689, 712, 728, 737,
 1116, 1117, 1304, 1308, 1323,
 1349, 1361, 1991, 1997, 2004, 2010
`\goalfraction` 6, 2111

H

`\hangingsymbol` 11, 1527, 1533
`\hb@xt@` .. 959, 966, 973, 996, 1003,
 1010, 1746, 1901, 1903, 1924, 1926
`\hsize` 855,
 895, 1746, 1901, 1903, 1924, 1926

I

`\if@filesw` 1671
`\if@firstcolumn` 1209, 1233, 1254, 1464
`\if@ledgroup` 593
`\if@nobreak` 820, 861
`\if@nopb` 1784, 1799
`\if@pb` 1783, 1800
`\if@pstarts` 1729, 1807, 1867, 1883
`\if@RTL` 684, 695, 707, 718
`\ifaraw@text` ... 1736, 1819, 1892, 1941
`\ifautopar` 847, 888
`\ifbypage@` 349
`\ifbypage@R` 140, 339, 1095
`\ifbypstart@` 568, 1759, 2136
`\ifbypstart@R` ... 140, 572, 1763, 2165
`\ifdim` 627, 631, 633,
 637, 1901, 1924, 1967, 2041, 2055
`\iffirst@linenum@out@R` 583, 587
`\ifinserthangingsymbol`
 1525, 1539, 2092
`\ifinserthangingsymbolR`
 1523, 1531, 1549, 2103
`\ifinstanzaL` 766, 766, 1526, 1538, 2090
`\ifinstanzaR` 766, 767, 1532, 1548, 2101
`\ifl@d@dash` 1333

- \ifl@d@elin 1335, 1336
 - \ifl@d@esl 1336
 - \ifl@d@pnum 1330, 1334
 - \ifl@d@ssub 1332
 - \ifl@d@pagefull 1911, 1933, 2033
 - \ifl@d@paging 11, 1537, 1547
 - \ifl@d@pairing 11, 81
 - \ifl@d@samelang 1659, 1737
 - \ifl@d@samepage 1896, 1920, 2033
 - \ifl@d@skipnumber 1186
 - \ifl@d@usedbabel 1657
 - \iflabelpstart 857, 897
 - \iflednopbinverse 2091, 2102
 - \ifledplinenum 1331
 - \ifledRcol 11, 172, 194,
198, 202, 206, 248, 263, 327,
336, 361, 375, 392, 409, 421,
429, 450, 495, 522, 531, 540,
611, 621, 628, 634, 640, 647,
655, 660, 664, 669, 679, 702,
723, 1344, 1382, 1396, 1409,
1423, 1437, 1485, 1498, 1694, 1704
 - \ifluatex 1499, 1510
 - \ifnoteschanged@ 91
 - \ifnumberedpar@ ... 830, 871, 903,
919, 1408, 1422, 1436, 1483, 1497
 - \ifnumbering 37, 145, 826, 900
 - \ifnumberingR ... 50, 78, 111, 867, 916
 - \ifnumberline 1019, 1036, 1185
 - \ifnumberpstart ... 848, 889, 912, 928
 - \ifnumequal 1452
 - \ifnumgreater 1460
 - \ifodd 1219, 1243,
1264, 1474, 1965, 1971, 1974, 1981
 - \ifpst@rtedL 32, 834
 - \ifpst@rtedR 32, 875
 - \ifpstartnum 1274, 1279
 - \ifpstartnumR 1229
 - \ifshiftedpstarts 5, 1900, 1923, 2112
 - \ifsidepstartnum 849, 890, 1231, 1252
 - \IfStrEq .. 1785, 1793, 2067, 2071,
2079, 2083, 2094, 2095, 2105, 2106
 - \ifsublines@ 226,
315, 360, 393, 400, 431, 440,
452, 459, 471, 505, 559, 561,
1020, 1037, 1102, 1189, 1384, 1388
 - \ifvbox 948, 985, 1823, 1826, 2125, 2154
 - \ifvmode 1351, 1363
 - \ifwritelinesL 2121, 2129
 - \ifwritelinesR 2122, 2158
 - \initnumbering@reg 45
 - \initnumbering@sectcmd 73
 - \initnumbering@sectcountR 74, 98
 - \insert@count 537, 617, 680,
703, 1416, 1430, 1444, 1492, 1519
 - \insert@countR 538, 613, 679,
702, 1412, 1426, 1440, 1488, 1508
 - \inserthangingsymbolfalse 957
 - \inserthangingsymbolL 966, 1523
 - \inserthangingsymbolR ... 1003, 1523
 - \inserthangingsymbolRfalse 994
 - \inserthangingsymbolRtrue 994
 - \inserthangingsymboltrue 957
 - \insertlines@listR
.... 84, 236, 250, 543, 1300, 1304
 - \inserts@list
836, 1415, 1429, 1443, 1491, 1518
 - \inserts@listR
.. 877, 1295, 1298, 1308, 1322,
1323, 1411, 1425, 1439, 1487, 1507
 - \istanzaLfalse 1778, 1956
 - \istanzaLtrue 777
 - \istanzaRfalse 1779, 1957
 - \istanzaRtrue 801
 - \interlinepenalty 1566
 - \itemcount@ 1450, 1452, 1457, 1460
- L**
- \l@d@nums 726, 729, 735, 738
 - \l@d@set 408, 655, 656
 - \l@dbbl@set@language 1668, 1691
 - \l@dbfnote 1482
 - \l@dc@maxchunks 842, 844,
883, 885, 1616, 1626, 1634, 1641
 - \l@dcalc@maxoftwo
..... 1872, 2020, 2145, 2174
 - \l@dcalc@minoftwo .. 1880, 1944, 2020
 - \l@dcalcnnum 1123
 - \l@dchecklang 1661, 1735
 - \l@dchset@num 282, 285, 408
 - \l@dcsnote 1405
 - \l@dcsnotetext 1459, 1462
 - \l@demptyd@ta 952, 989
 - \l@dend@stuff ... 48, 70, 116, 130, 138
 - \l@dgetline@margin 170
 - \l@dgetsidenote@margin 1394
 - \l@dld@ta 962, 999,
1210, 1222, 1234, 1246, 1255, 1267
 - \l@dleftbox
.. 933, 958, 973, 1747, 1901, 1903

- \l@dlinenumR [218](#)
- \l@dlsn@te [965](#), [1002](#)
- \l@dlsnote [1405](#)
- \l@dmake@labels [1367](#)
- \l@dmake@labelsR [1355](#), [1371](#)
- \l@dminpagelines
.... [1843](#), [1881](#), [1945](#), [2042](#), [2056](#)
- \l@dnumstartsL . [41](#), [841](#), [842](#), [844](#),
[846](#), [1620](#), [1652](#), [1717](#), [1718](#),
[1775](#), [1810](#), [1852](#), [1853](#), [1954](#), [2134](#)
- \l@dnumstartsR . [54](#), [882](#), [883](#), [885](#),
[887](#), [1620](#), [1653](#), [1717](#), [1718](#),
[1776](#), [1813](#), [1852](#), [1853](#), [1955](#), [2163](#)
- \l@doldbbl@set@language [1690](#)
- \l@doldselectlanguage [1689](#), [1693](#), [1698](#)
- \l@dpagfullfalse
.... [2033](#), [2069](#), [2074](#), [2081](#), [2086](#)
- \l@dpagfulltrue
.... [2033](#), [2068](#), [2073](#), [2080](#), [2085](#)
- \l@dpagingfalse [13](#), [745](#), [758](#)
- \l@dpagingtrue [753](#)
- \l@dpairingfalse [11](#), [747](#), [757](#)
- \l@dpairingtrue [744](#), [752](#)
- \l@dpscL [948](#), [953](#), [1622](#), [1654](#), [1726](#),
[1732](#), [1773](#), [1810](#), [1823](#), [1862](#),
[1868](#), [1873](#), [1876](#), [1884](#), [1952](#),
[2125](#), [2127](#), [2134](#), [2145](#), [2147](#), [2149](#)
- \l@dpscR [985](#), [990](#), [1623](#), [1655](#),
[1727](#), [1733](#), [1774](#), [1813](#), [1826](#),
[1863](#), [1869](#), [1877](#), [1885](#), [1953](#),
[2154](#), [2156](#), [2163](#), [2174](#), [2176](#), [2178](#)
- \l@drd@ta [966](#), [1003](#),
[1212](#), [1220](#), [1236](#), [1244](#), [1257](#), [1265](#)
- \l@drightbox
.. [933](#), [995](#), [1010](#), [1749](#), [1924](#), [1926](#)
- \l@drsn@te [967](#), [1004](#)
- \l@drsnote [1405](#)
- \l@dsamelangfalse [1659](#), [1662](#)
- \l@dsamelangtrue [1659](#), [1665](#)
- \l@dsamepagfalse
.... [2033](#), [2068](#), [2073](#), [2080](#), [2085](#)
- \l@dsamepagtrue
.... [2033](#), [2069](#), [2074](#), [2081](#), [2086](#)
- \l@dsetupmaxlinecounts .. [1633](#), [1650](#)
- \l@dsetuprawboxes [1625](#), [1649](#)
- \l@dskipnumberfalse [1187](#)
- \l@dskipnumbertrue [1083](#)
- \l@dunhbox@line [966](#), [1003](#)
- \l@dusedbabelfalse [1657](#), [1686](#)
- \l@dusedbabeltrue [1657](#), [1688](#)
- \l@duselanguage
.... [1678](#), [1741](#), [1743](#), [1894](#), [1919](#)
- \l@dzeromaxlinecounts ... [1633](#), [1651](#)
- \l@dzeropenalties [906](#), [922](#), [1721](#), [1857](#)
- \l@prev@nopbR [61](#), [2184](#)
- \l@prev@pbR [60](#), [2183](#)
- \l@pscL [1622](#)
- \l@pscR [1622](#)
- \label@refs
[1347](#), [1349](#), [1355](#), [1359](#), [1361](#), [1367](#)
- \labelref@list [1358](#), [1361](#), [1389](#)
- \labelref@listR [1341](#), [1346](#), [1349](#), [1385](#)
- \language name .. [1669](#), [1670](#), [1672](#)–[1675](#)
- \last@page@num [347](#), [353](#)
- \last@page@numR [333](#)
- \lastbox [955](#), [992](#)
- \lastskip [626](#), [632](#)
- \Lcolwidth [5](#), [6](#), [15](#), [754](#), [855](#), [959](#), [973](#)
- \led@err@BadLeftRightPstarts ...
..... [23](#), [1718](#), [1853](#)
- \led@err@LeftOnRightPage ... [26](#), [1975](#)
- \led@err@LineationInNumbered ... [146](#)
- \led@err@NumberingNotStarted ... [95](#)
- \led@err@numberingShouldHaveStarted
..... [118](#)
- \led@err@NumberingStarted [38](#), [51](#)
- \led@err@PendNoPstart [904](#), [920](#)
- \led@err@PendNotNumbered ... [901](#), [917](#)
- \led@err@PstartInPstart ... [831](#), [872](#)
- \led@err@PstartNotNumbered . [827](#), [868](#)
- \led@err@RightOnLeftPage ... [26](#), [1982](#)
- \led@err@TooManyPstarts [20](#), [843](#), [884](#)
- \led@mess@NotesChanged [92](#)
- \led@mess@SectionContinued
..... [114](#), [128](#), [136](#)
- \led@nopbnumR [2188](#), [2189](#)
- \led@nopbR [2187](#), [2189](#)
- \led@pb@setting
.... [1785](#), [1793](#), [2067](#), [2071](#),
[2079](#), [2083](#), [2094](#), [2095](#), [2105](#), [2106](#)
- \led@pbnumR [2186](#), [2189](#)
- \led@pbR [2185](#), [2189](#)
- \led@warn@BadAction [1085](#)
- \led@warn@BadAdvancelineLine [378](#), [384](#)
- \led@warn@BadAdvancelineSubline .
..... [364](#), [370](#)
- \led@warn@BadLineation [163](#)
- \led@warn@BadSetline [645](#)
- \led@warn@BadSetlinenum [653](#)
- \led@warn@DuplicateLabel [1373](#)

- \ledllfill 966, 1003
 - \lednopb 797
 - \lednopbnum 2094, 2185
 - \lednopbnumR 2105, 2185
 - \lednopbR 797, 2187
 - \ledpb 796
 - \ledpbnum 2095
 - \ledpbnumR 2106, 2185
 - \ledpbR 796, 2185
 - \ledRcolfalse 14, 769, 803
 - \ledRcoltrue 787
 - \ledrlfill 966, 1003
 - \ledsavedprintlines 9, 1328
 - \ledstrutL 1901, 1903, 1960
 - \ledstrutR 1924, 1926, 1960
 - \ledthegoal 2041, 2055, 2111
 - \leftlinenumR 218, 1210, 1222
 - \leftpstartnumL 1229
 - \leftpstartnumR 1229
 - Leftside (environment) 6, 768
 - \Leftsidehook 775, 781
 - \Leftsidehookend 780, 781
 - \line@list 733, 737
 - \line@list@stuff 47, 129
 - \line@list@stuffR .. 69, 115, 137, 585
 - \line@listR . 87, 236, 249, 561, 724, 728
 - \line@margin 175, 1239
 - \line@marginR 168, 1215, 1260
 - \line@num . 350, 382, 383, 385, 403, 414, 415, 443, 568, 1043, 1387, 2139
 - \line@numR 62, 225, 232, 287, 321, 340, 376, 377, 379, 396, 410, 411, 434, 554, 558, 572, 1026, 1096, 1105, 1194, 1196, 1198, 1199, 1383, 1460, 2168
 - \lineation 798
 - \lineationR 144, 798
 - \linenum@out 607, 616, 624, 629, 635, 641, 648, 656, 661, 665, 1357, 1760, 1834, 2014
 - \linenum@outR 582, 588, 590, 594, 595, 597, 598, 601, 602, 609, 612, 622, 628, 634, 640, 647, 655, 660, 664, 669, 1345, 1764, 1839, 2017, 2185–2188
 - \linenumberlist 1195, 1199
 - \linenumincrement 6, 189
 - \linenumincrement* 6, 189
 - \linenummargin 168
 - \linenumr@p 1331, 1335, 1383, 1387
 - \linenumrepR 215, 225
 - \linenumsep 220, 222, 1276, 1279, 1288, 1291
 - \linesinpar@listL 241, 257, 569, 1988, 1991
 - \linesinpar@listR 241, 253, 573, 1994, 1997
 - \linesonpage@listL 258, 577, 2001, 2004
 - \linesonpage@listR 254, 580, 2007, 2010
 - \list@clear 249–254, 257, 258, 260, 836, 877
 - \list@clearing@reg 256
 - \list@create 236–239, 241–243, 1295, 1341
 - \listcsxadd 355, 2189–2192
 - \lock@disp 1155, 1159, 1164
 - \lock@off 492, 493, 501, 664, 665
 - \lock@on 660, 661
- M**
- \managestanza@modulo 1592
 - \maxchunks 4, 1616
 - \maxlinesinpar@list 241, 260
 - \memorydump 8, 774, 792
 - \memorydumpL 123, 774
 - \memorydumpR 123, 792
 - \message 46, 68
 - \multiply 1129
- N**
- \n@num 529, 669
 - \n@num@reg 535
 - \namebox 948, 953, 985, 990, 1600, 1823, 1826, 2125, 2154
 - \NeedsTeXFormat 2
 - \new@lineL 606, 966
 - \new@lineR 608, 1003
 - \newbox 809, 933, 934, 1601
 - \newcounter 98–101, 180, 182, 184, 186, 812, 813, 815, 816
 - \newif 5, 12, 33, 140, 141, 583, 766, 767, 1283, 1523, 1657, 1659, 1783, 1784, 1807, 1819, 2033, 2035, 2121, 2122
 - \newnamebox 1600, 1628, 1629
 - \newnamecount 1611, 1636
 - \newwrite 582
 - \next@absline 1786, 1788, 1790, 2072–2074

- `\next@abslineR` 1787, 1789, 1791, 2084–2086
 - `\next@action` 277
 - `\next@actionline` 274, 276
 - `\next@actionlineR` 266, 268, 1054, 1092, 1114, 1116
 - `\next@actionR` 269, 1055, 1093, 1094, 1099, 1100, 1108, 1117
 - `\next@insert` 837
 - `\next@insertR` 878, 1299, 1302, 1304, 1307, 1311
 - `\next@page@num` 354, 426
 - `\next@page@numR` 66, 290, 292, 344, 423
 - `\no@expands` 677, 700
 - `\normal@page@breakR` 59
 - `\normal@pars` 80, 840, 881
 - `\normalbfnoteX` 1496
 - `\noteschanged@true` 85, 88, 725, 734, 1301
 - `\num@lines` 907, 1722, 1858
 - `\num@linesR` 808, 923, 1723, 1859
 - `\numberedpar@true` 856, 896
 - `\numberingRfalse` 79
 - `\numberingRtrue` 56, 109, 133
 - `\numberingtrue` 43, 125
 - `\numberpstartfalse` 9
 - `\numberpstarttrue` 9
 - `\numdef` 1450, 1457, 1786, 1787, 2072, 2084
 - `\numgdef` 2093, 2104
 - `\numlabfont` 225
 - `\numpagelinesL` 1843, 1899, 1912, 1916, 2042, 2095
 - `\numpagelinesR` 1843, 1922, 1934, 1938, 2056, 2106
- O**
- `\oldchapter` 750, 759
 - `\oldstanza` 776, 777, 779, 800, 801, 804
 - `\one@line` 953, 955, 966
 - `\one@lineR` 808, 990, 992, 1003
 - `\openout` 590, 595, 598, 602
- P**
- `\p@pstartL` 858
 - `\p@pstartR` 898
 - `\page@action` 291, 420, 548
 - `\page@num` 272, 352, 1241, 1472
 - `\page@numR` 245, 264, 342, 553, 558, 1094, 1217, 1262, 1460
 - `\pagebreak` 1800
 - `\pagegoal` 2119
 - `\Pages` 6, 1847
 - `pages` (environment) 5, 743
 - `\pagetotal` 1967, 2041, 2055
 - `pairs` (environment) 5, 743
 - `\par@line` 908, 1724, 1860
 - `\par@lineR` 808, 924, 1725, 1861
 - `\pausenumbering` 790
 - `\pausenumberingR` 108, 790
 - `\pend` 7, 773, 795, 832, 1579
 - `\pendL` 773, 900
 - `\pendR` 795, 873, 916
 - `\prev@abslineverse` 2093–2095, 2104–2106
 - `\prev@nopbR` 2183
 - `\prev@pbR` 2183
 - `\prevgraf` 907, 923, 1722, 1723, 1858, 1859
 - `\printlines` 1339
 - `\printlinesR` 9, 1328
 - `\ProcessOptions` 10
 - `\protected@csxdef` 1505, 1516
 - `\protected@edef` 857, 897
 - `\protected@write` ... 1354, 1366, 1672
 - `\ProvidesPackage` 3
 - `\pst@rtedLfalse` 32, 42
 - `\pst@rtedLtrue` 126, 838
 - `\pst@rtedRfalse` 34, 55, 82
 - `\pst@rtedRtrue` 112, 134, 879
 - `\pstart` 7, 21, 25, 771, 794, 1581
 - `\pstartL` 771, 811
 - `\pstartnumfalse` 1276, 1281
 - `\pstartnumRfalse` 1288, 1293
 - `\pstartnumRtrue` 1284, 1731, 2173
 - `\pstartnumtrue` 1730, 2144
 - `\pstartR` 794, 811
- R**
- `\Rcolwidth` 5, 6, 15, 755, 895, 996, 1010
 - `\read@linelist` 247, 586
 - `\rem@inder` 1199, 1201–1203
 - `\resetprevline@` 1761, 1765, 2140, 2169
 - `\resumenumbering` 791
 - `\resumenumberingR` 108, 791
 - `\rightlinenumR` 218, 1212, 1220
 - `\rightpstartnumL` 1229
 - `\rightpstartnumR` 1229
 - `Rightside` (environment) 6, 786
 - `\Rightsidehook` 781, 799
 - `\Rightsidehookend` 781, 805

\rlap 1212, 1220, 1236, 1244, 1257, 1265
 \Rlineflag 8, 213, 225, 1331, 1335, 1375
 \rule 1803

S

\sc@n@list 1200, 1202
 \secdef 764
 \section@num 44, 46, 47, 127–129
 \section@numR
 . 30, 57, 68, 69, 113–115, 135–137
 \select@language ... 1670, 1672–1675
 \selectlanguage 1678
 \set@line 678, 701, 722
 \set@line@action
 284, 389, 398, 405, 428, 550
 \setl@dlp@rbox 1465, 1477
 \setl@drp@rbox 1467, 1475
 \setline 643
 \setlinenum 651
 \setnamebox 846, 887, 1600
 \setprintlines 1329
 \shiftedpstartfalse 7
 \shiftedpstarttrue 6, 8, 9
 \shiftedversesfalse 7
 \shiftedversetrue 6
 \showlemma 687, 710
 \sidenote@margin 1399, 1403
 \sidenote@marginR 1392, 1470
 \sidenotecontent@
 1449, 1454, 1455, 1465, 1467, 1477
 \sidenotecontent@t 1475
 \sidenotemargin 1392
 \sidenotesep 1455
 \skip@lockoff 493, 501
 \skipnumbering 9, 668
 \skipnumbering@reg 672
 \smash 1803
 \splittopskip 950, 987
 \stanza ... 776, 777, 779, 800, 801, 804
 \stanza@count 1561, 1575, 1587
 \stanza@hang 1563, 1595
 \stanza@modulo 1561, 1590
 \stanzaindentbase
 1541, 1551, 1588, 1591
 \startlock 659
 \startstanzahook 1559
 \startsub 626
 \sub@action 300, 449, 549
 \sub@change 67, 294, 295, 301
 \sub@lock 1038

\sub@lockR 64, 309, 311, 313,
 316, 467, 473, 474, 476, 477,
 507, 508, 510, 1021, 1075, 1077,
 1078, 1080, 1136, 1176, 1178, 1180
 \sub@off 634, 635
 \sub@on 628, 629
 \subline@num 227, 350, 368,
 369, 371, 401, 441, 1039, 1044, 1388
 \subline@numR 228,
 232, 317, 321, 340, 362, 363,
 365, 394, 432, 555, 559, 1022,
 1027, 1096, 1103, 1190, 1191, 1384
 \sublinenumincrement 6, 189
 \sublinenumincrement* 6, 189
 \sublinenumr@p . 1332, 1336, 1384, 1388
 \sublinenumrepR 215, 228
 \sublines@false 65, 298, 1065
 \sublines@true 296, 1063
 \sublock@disp 1138, 1142, 1147
 \symplinenum 1331
 \sza@penalty 1570, 1574

T

\textwidth 16, 18, 754, 755
 \theledlanguageL 1663, 1678, 1741, 1894
 \theledlanguageR 1663, 1678, 1743, 1919
 \thepage 607, 609, 1355, 1367
 \thepstart 772, 793
 \thepstartL
 . 9, 772, 814, 851, 858, 1275, 1280
 \thepstartR
 . 9, 793, 817, 891, 898, 1287, 1292
 \thr@@ .. 476, 485, 508, 515, 1070, 1078
 \topskip 1967

U

\unhbox 1605,
 1747, 1749, 1901, 1903, 1924, 1926
 \unhnamebox 1600
 \unvbox 955, 992, 1607
 \unvnamebox 1600
 \usernamecount
 . 1562, 1569, 1611, 1643, 1873,
 2127, 2145, 2147, 2156, 2174, 2176

V

\value 835, 876, 1586,
 1715, 1716, 1848, 1849, 2137, 2166
 \vbadness 949, 986
 \vbfnoteX 1506, 1517

<code>\vbox</code>	846, 887	<code>\xifinlistcs</code>	1788–
<code>\vl@dbfnote</code>	1486, 1490		1791, 1794–1797, 2068, 2069,
<code>\vl@dcnote</code>	1438, 1442		2073, 2074, 2080, 2081, 2085, 2086
<code>\vl@dlsnote</code>	1410, 1414	<code>\xpg@main@language</code>	1708, 1709
<code>\vl@drsnote</code>	1424, 1428	<code>\xpg@set@language</code>	1703, 1707
<code>\vsplit</code>	953, 990	<code>\xright@appenditem</code>	
		 422, 423, 425, 426, 430,
			437, 439, 446, 451, 453, 455,
			458, 460, 462, 470, 472, 481,
			504, 506, 513, 532, 533, 543,
			557, 569, 573, 577, 580, 1383,
			1387, 1410, 1414, 1424, 1428,
			1438, 1442, 1486, 1490, 1506, 1517
W			
<code>\wd</code>	966, 1003		
<code>\WithSuffix</code>	209–212		
<code>\writtenlinesLfalse</code>	1864, 2135		
<code>\writtenlinesLtrue</code>	2132		
<code>\writtenlinesRfalse</code>	1865, 2164		
<code>\writtenlinesRtrue</code>	2161		
X			
<code>\x@lemma</code>	689–691, 712–714	Z	
		<code>\zz@@@</code>	1347, 1359

Change History

v0.1		eledmac 0.15).	44
General: First public release	1	<code>\Columns</code> : Line numbering by	
v0.10		pstart.	62
General: <code>\edlabel</code> commands on		<code>\get@nextboxR</code> : Change <code>\get@nextboxL</code>	
the right side are now correctly		and <code>\get@nextboxR</code> to allow to	
indicated.	1	disable line numbering (like in	
<code>\edlabel</code> commands which start		eledmac 0.15).	72
a paragraph are now put in the		Pstart number can be printed in	
right place.	1	side	72
v0.11		v0.12	
General: Change <code>\do@lineL</code> and		General: New new management of	
<code>\do@lineR</code> to allow line num-		hangingsymbol insertion, pre-	
bering by pstart (like in eledmac		venting undesirable insertions.	54
0.15).	39	v0.2	
Lineation can be by pstart (like		General: Added section of babel re-	
in eledmac 0.15).	16	lated code	58
New management of hang-		Fix babel problems	1
ingsymbol insertion, preventing		<code>\Columns</code> : Added <code>\l@dchecklang</code>	
undesirable insertions.	54	and <code>\l@duselanguage</code> to	
Prevent shift of column separator		<code>\Columns</code>	61
when a verse is hanged	55	<code>\Pages</code> : Added <code>\l@duselanguage</code>	
<code>\affixline@numR</code> : Changed		to <code>\Pages</code>	66
<code>\affixline@numR</code> to allow to		v0.3	
disable line numbering (like in		General: Added <code>\do@lineLhook</code>	

and <code>\do@lineRhook</code>	40	v0.3.b	
Reorganize for ledarab	1	General: Improved parallel page balancing	1
<code>\affixline@numR</code> : Changed <code>\affixline@numR</code> to match new eledmac	44	v0.3.c	
<code>\do@actions@nextR</code> : Used <code>\do@actions@fixedcode</code> in <code>\do@actionsR</code>	42	General: Compatibilty with Poly- glossia	1
<code>\do@lineL</code> : Added <code>\do@lineLhook</code> to <code>\do@lineL</code>	39	v0.3a	
Simplified <code>\do@lineL</code> by using macros for some common code	39	<code>\line@marginR</code> : Don't just set <code>\line@marginR</code> in <code>\linenummargin</code>	17
<code>\do@lineR</code> : Changed <code>\do@lineR</code> similarly to <code>\do@lineL</code>	40	v0.3b	
Leftside: Added hooks into Left- side environment	34	<code>\Pages</code> : Added <code>\l@dminpagelines</code> calculation for succeeding page pairs	67
<code>\flag@end</code> : Removed extraneous spaces from <code>\flag@end</code>	30	v0.4	
<code>\ifledRcol</code> : Moved <code>\ifl@dpairing</code> to eledmac	13	General: No more ledparpatch. All patches are now in the main file.	1
<code>\ifpst@rtedR</code> : Moved <code>\ifpst@rtedL</code> to eledmac	14	v0.5	
<code>\l@dlinenumR</code> : Simplified <code>\leftlinenumR</code> and <code>\rightlinenumR</code> by introducing <code>\l@dlinenumR</code>	19	General: Corrections about <code>\section</code> and other titles in numbered sections	1
<code>\l@dnumpstartsR</code> : Moved <code>\l@dnumpstartsL</code> to eledmac .	57	v0.6	
<code>\ledsavedprintlines</code> : Simpli- fied <code>\printlinesR</code> by using <code>\setprintlines</code>	49	General: Be able to us <code>\chapter</code> in parallel pages.	1
<code>\ledstrutR</code> : Added <code>\ledstrutL</code> and <code>\ledstrutR</code>	68	v0.7	
<code>\normalbfnoteX</code> : Removed extraneous spaces from <code>\normalbfnoteX</code>	54	General: Option 'shiftedverses' which make there is no blank between two parallel verses with inequal length.	1
<code>\Pages</code> : Added <code>\ledstrutL</code> to <code>\Pages</code>	66	v0.8	
Added <code>\ledstrutR</code> to <code>\Pages</code> .	66	General: Possibility to have a sym- bol on each hanging of verses, like in the french typogra- phy. Redefine the commande <code>\hangingsymbol</code> to define the character.	1
<code>\Rightsidehookend</code> : Added <code>\Leftsidehook</code> , <code>\Leftsidehookend</code> , <code>\Rightsidehook</code> and <code>\Rightsidehookend</code>	35	v0.9	
<code>\sublinenumrepR</code> : Added <code>\linenumrepR</code> and <code>\sublinenumrepR</code>	19	General: Possibility to number <code>\pstart</code>	9
v0.3.a		Possibility to number the pstart with the commands <code>\numberpstarttrue</code>	1
General: Minor <code>\linenummargin</code> fix	1	<code>\ifledRcol</code> : Moved <code>\iflledRcol</code> and <code>\ifnumberingR</code> to eledmac	13
		v0.9.1	
		General: The numbering of the pstarts restarts on each <code>\beginnumbering</code>	1

v0.9.2	General: Debug : with <code>\Columns</code> , the hanging indentation now runs on the left columns and the hanging symbol is shown only when <code>\stanza</code> is used. 1	<code>\l@dcnote</code> : Debug on the left notes of the right column. . . . 52
v0.9.3	General: <code>\thepstartL</code> and <code>\thepstartR</code> use now <code>\bfseries</code> and not <code>\bf</code> , which is deprecated and makes con- flicts with memoir class. 1	v1.3.4 <code>\l@dcnote</code> : Allow to use com- mands in sidenotes, like it was introduced by eledmac 1.0. . . 52
v1.0	General: Compatibility with eled- mac. Change name to eledpar. . 1	v1.3.5 <code>\normalbfnoteX</code> : Allows one to redefine <code>\thefootnoteX</code> with alph when some packages are loaded. 54
v1.0.1	General: Correction on <code>\numberonlyfirstinline</code> with lineation by pstart or by page. 1	v1.4 General: Added <code>\do@insidelineLhook</code> and <code>\do@insidelineRhook</code> . . . 40
v1.1	General: <code>Shiftedverses</code> becomes <code>shiftedpstarts</code> 1	v1.4.1 <code>\normalbfnoteX</code> : Fix bug with nor- mal familiar footnotes when mixing RTL and LTR text. . . 54
v1.1.1	<code>\pstartR</code> : Add <code>\labelpstarttrue</code> (from eledmac). 36	<code>\astanza</code> : Enable the use of stan- zaindent repetition within as- tanza environment. 55
v1.1.1.1	<code>\pstartR</code> : Correct <code>\pstartR</code> bug in- troduced by 1.1. 36	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. 29
v1.1.1.2	<code>\affixside@noteR</code> : Remove spuri- ous space between line number and line content 53	v1.4.3 General: Corrects a false hanging verse when a verse is exactly the length of a line. 1
v1.2	General: Support for <code>\led<section></code> commands in parallel texts. . . . 1	<code>\inserthangingsymbolR</code> : Hang verse is now not automatically flush right. 54
v1.2.1	<code>\initnumbering@sectcountR</code> : For the right section, the counter is defined only once. 15	<code>\pendL</code> : Spurious spaces in <code>\pendL</code> . 38
v1.3	General: Manage RTL language. . 32	<code>\pendR</code> : Spurious spaces in <code>\pstartR</code> 39
v1.3.2	General: Debug with some classes. 1	<code>\pstartR</code> : Spurious spaces in <code>\pstartL</code> and <code>\pstartR</code> 36
v1.3.3	<code>\l@dbfnote</code> : Spurious space with footnote in right column. . . . 53	v1.5.0 General: Add, as in eledmac, fea- tures to manage page breaks. . . 1
		<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. 18