

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	3
2.1	General	4
3	Parallel columns	5
4	Facing pages	5
5	Left and right texts	6
6	Numbering text lines and paragraphs	7

*This file (**eledpar.dtx**) has version number v1.4.2, last revised 2013/08/21.

[†]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	31
13 Parallel environments	33
14 Paragraph decomposition and reassembly	35
14.1 Boxes, counters, <code>\pstart</code> and <code>\pend</code>	35
14.2 Processing one line	39
14.3 Line and page number computation	41
14.4 Line number printing	43
14.5 Pstart number printing in side	45
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	49
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	63
25 The End	71
References	72
Index	72
Change History	80

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 25); 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 *pdf_latex* or *x_latex*. 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** environment. 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.

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

\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

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

`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 `eledpar` with the option `shiftedpstarts`.

6 Numbering text lines and paragraphs

`\beginnumbering` Each section of numbered text must be preceded by `\beginnumbering` and fol-
`\endnumbering`

lowed 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}}
```

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

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

7 Verse

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

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

If you get an error message along the lines of 'Missing number, treated as zero `\sza@00`' it is because you have forgotten to use `\setstanzaindents` to set the stanza indents.

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

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

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

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

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

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

\end{astanza}
...
```

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

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

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}[2013/08/21 v1.4.2 eledmac extension for parallel texts]
4
```

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

```
\ifshiftedpstarts
```

```

5 \newif\ifshiftedpstarts
6 \let\shiftedversestrue\shiftedpstartstrue
7 \let\shiftedversesfalse\shiftedpstartsfalse
8 \DeclareOption{shiftedverses}{\shiftedpstartstrue}
9 \DeclareOption{shiftedpstarts}{\shiftedpstartstrue}
10 \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 \qquad 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 \global\line@numR \z@
60 \global\@lockR \z@
61 \global\sub@lockR \z@
62 \global\sublines@false
63 \global\let\next@page@numR\relax
64 \global\let\sub@change\relax
65 \message{Section \the\section@numR R }%
66 \line@list@stuffR{\jobname.\extensionchars\the\section@numR R}%
67 \l@dend@stuff
68 \setcounter{pstartR}{1}
69 \begingroup
70 \initnumbering@sectcmd
71 \initnumbering@sectcountR
72 }
```

73

`\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.

```

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

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

```

95 \newcounter{chapterR}
96 \newcounter{sectionR}
97 \newcounter{subsectionR}
98 \newcounter{subsubsectionR}
99 \newcommand{\initnumbering@sectcountR}{
100 \let\c@chapter\c@chapterR
101 \let\c@section\c@sectionR
102 \let\c@subsection\c@subsectionR
103 \let\c@subsubsection\c@subsubsectionR
104 }
```

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

```

\resumenumberingR 105 \newcommand*{\pausenumberingR}{%
106   \endnumberingR\global\numberingRtrue}
107 \newcommand*{\resumenumberingR}{%
108   \ifnumberingR
109     \global\pst@rtedRtrue
```

```

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

```

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

```

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

```

11 Line counting

11.1 Choosing the system of lineation

M 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.

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

<code>\ifbypstart@R</code>	
<code>\bypstart@Rtrue</code>	
<code>\bypstart@Rfalse</code>	• line-of-page : <code>bypstart@R = false</code> and <code>bypage@R = true</code> .
<code>\ifbypage@R</code>	
<code>\bypage@Rtrue</code>	• line-of-pstart : <code>bypstart@R = true</code> and <code>bypage@R = false</code> .
<code>\bypage@Rfalse</code>	

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

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

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

`\linenummargin` `\linenummargin{<word>}` 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.

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

```

173   \fi
174   \fi}}

```

By default put right text numbers at the right.

```

175 \line@marginR=\@ne
176

```

`\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.

```

177 \newcounter{firstlinenumR}
178   \setcounter{firstlinenumR}{5}
179 \newcounter{linenumincrementR}
180   \setcounter{linenumincrementR}{5}

```

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

```

181 \newcounter{firstsublinenumR}
182   \setcounter{firstsublinenumR}{5}
183 \newcounter{sublinenumincrementR}
184   \setcounter{sublinenumincrementR}{5}
185

```

`\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 `\provide`ing them.

```

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

```

```

\lRlineflag This is appended to the line numbers of right text.
207 \newcommand*{\lRlineflag}{R}
208

\linenumrepR \linenumrepR{<ctr>} typesets the right line number <ctr>, and similarly \sublinenumrepR
\sublinenumrepR for subline numbers.
209 \newcommand*{\linenumrepR}[1]{\@arabic{#1}}
210 \newcommand*{\sublinenumrepR}[1]{\@arabic{#1}}
211

\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.
212 \newcommand*{\leftlinenumR}{%
213 \l@dlinenumR
214 \kern\linenumsep}
215 \newcommand*{\rightlinenumR}{%
216 \kern\linenumsep
217 \l@dlinenumR}
218 \newcommand*{\l@dlinenumR}{%
219 \numlabfont\linenumrepR{\line@numR}\lRlineflag%
220 \ifsublines@
221 \ifnum\subline@num>\z@
222 \unskip\fullstop\sublinenumrepR{\subline@numR}%
223 \fi
224 \fi}
225

```

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.
226 \newcount\line@numR
227 \newcount\subline@numR
228 \newcount\absline@numR
229

\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:

```

```

230 \list@create{\line@listR}
231 \list@create{\insertlines@listR}
232 \list@create{\actionlines@listR}
233 \list@create{\actions@listR}
234

```

`\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.

```

235 \list@create{\linesinpar@listL}
236 \list@create{\linesinpar@listR}
237 \list@create{\maxlinesinpar@list}
238

```

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

```

239 \newcount\page@numR
240

```

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.

```

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

```

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

```

242 \ifledRcol
243 \list@clear{\line@listR}%
244 \list@clear{\insertlines@listR}%
245 \list@clear{\actionlines@listR}%
246 \list@clear{\actions@listR}%
247 \list@clear{\linesinpar@listR}%
248 \list@clear{\linesonpage@listR}
249 \else
250 \list@clearing@reg
251 \list@clear{\linesinpar@listL}%
252 \list@clear{\linesonpage@listL}%
253 \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).

```

254 \list@clear{\maxlinesinpar@list}

```

Now get the file and interpret it.

```

255 \get@linelistfile{#1}%
256 \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.

```

257 \ifledRcol
258   \global\page@numR=\m@ne
259   \ifx\actionlines@listR\empty
260     \gdef\next@actionlineR{1000000}%
261   \else
262     \gl@p\actionlines@listR\to\next@actionlineR
263     \gl@p\actions@listR\to\next@actionR
264   \fi
265 \else
266   \global\page@num=\m@ne
267   \ifx\actionlines@list\empty
268     \gdef\next@actionline{1000000}%
269   \else
270     \gl@p\actionlines@list\to\next@actionline
271     \gl@p\actions@list\to\next@action
272   \fi
273 \fi}
274

```

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.

```

275 \newcommand{\@l@regR}{%
276   \ifx\l@dchset@num\relax \else
277     \advance\absline@numR \@ne
278     \set@line@action
279     \let\l@dchset@num\relax
280     \advance\absline@numR \m@ne
281     \advance\line@numR \m@ne%    % do we need this?
282   \fi
283   \advance\absline@numR \@ne
284   \ifx\next@page@numR\relax \else
285     \page@action
286     \let\next@page@numR\relax
287   \fi
288   \ifx\sub@change\relax \else

```

```

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

```

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

```

\fix@page 327 \newcount\last@page@numR
328 \last@page@numR=-10000
329 \renewcommand*{\fix@page}[1]{%
330 \ifledRcol
331 \ifnum #1=\last@page@numR
332 \else
333 \ifbypage@R
334 \line@numR \z@ \subline@numR \z@
335 \fi
336 \page@numR=#1\relax

```

```

337     \last@page@numR=#1\relax
338     \def\next@page@numR{#1}%
339     \fi
340 \else
341     \ifnum #1=\last@page@num
342     \else
343         \ifbypage@
344             \line@num \z@ \subline@num \z@
345         \fi
346         \page@num=#1\relax
347         \last@page@num=#1\relax
348         \def\next@page@num{#1}%
349     \fi
350 \fi}
351

```

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

```

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

```

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

```

384 \renewcommand*{\@set}[1]{%
385   \ifledRcol
386     \ifsublines@
387       \subline@numR=#1\relax
388     \else
389       \line@numR=#1\relax
390     \fi
391     \set@line@action
392   \else
393     \ifsublines@
394       \subline@num=#1\relax
395     \else
396       \line@num=#1\relax
397     \fi
398     \set@line@action
399   \fi}
400
```

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

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

```

401 \renewcommand*{\l@d@set}[1]{%
402   \ifledRcol
403     \line@numR=#1\relax
404     \advance\line@numR \@ne
405     \def\l@dchset@num{#1}
406   \else
407     \line@num=#1\relax
408     \advance\line@num \@ne
409     \def\l@dchset@num{#1}
410   \fi}
411 \let\l@dchset@num\relax
412
```

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

```

413 \renewcommand*{\page@action}{%
414   \ifledRcol
415     \xright@appenditem{\the\absline@numR}\to\actionlines@listR
416     \xright@appenditem{\next@page@numR}\to\actions@listR
417   \else
418     \xright@appenditem{\the\absline@num}\to\actionlines@list
419     \xright@appenditem{\next@page@num}\to\actions@list
420   \fi}

```

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


```

421 \renewcommand*{\set@line@action}{%
422   \ifledRcol
423     \xright@appenditem{\the\absline@numR}\to\actionlines@listR
424     \ifsublines@
425       \@l@dtmpcnta=-\subline@numR
426     \else
427       \@l@dtmpcnta=-\line@numR
428     \fi
429     \advance\@l@dtmpcnta by -5000\relax
430     \xright@appenditem{\the\@l@dtmpcnta}\to\actions@listR
431   \else
432     \xright@appenditem{\the\absline@num}\to\actionlines@list
433     \ifsublines@
434       \@l@dtmpcnta=-\subline@num
435     \else
436       \@l@dtmpcnta=-\line@num
437     \fi
438     \advance\@l@dtmpcnta by -5000\relax
439     \xright@appenditem{\the\@l@dtmpcnta}\to\actions@list
440   \fi}
441

```

\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.

```

442 \renewcommand*{\sub@action}{%
443   \ifledRcol
444     \xright@appenditem{\the\absline@numR}\to\actionlines@listR
445     \ifsublines@
446       \xright@appenditem{-1001}\to\actions@listR
447     \else
448       \xright@appenditem{-1002}\to\actions@listR
449     \fi
450   \else
451     \xright@appenditem{\the\absline@num}\to\actionlines@list
452     \ifsublines@
453       \xright@appenditem{-1001}\to\actions@list
454     \else
455       \xright@appenditem{-1002}\to\actions@list
456     \fi
457   \fi}
458

```

\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.

```

459 \newcount\@lockR
460 \newcount\sub@lockR
461
462 \newcommand*{\do@lockonR}{%

```

```

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

\lock@off \lock@off adds an entry to the action-code list to turn line number locking off.
\do@lockoff 494
\do@lockoffR 495
\skip@lockoff 496 \newcommand{\do@lockoffR}{\%
497 \xright@appenditem{\the\absline@numR}\to\actionlines@listR
498 \ifsublines@
499 \xright@appenditem{-1006}\to\actions@listR
500 \ifnum\sub@lockR=\tw@
501 \sub@lockR \thr@@
502 \else
503 \sub@lockR \z@
504 \fi
505 \else
506 \xright@appenditem{-1004}\to\actions@listR
507 \ifnum\@lockR=\tw@
508 \@lockR \thr@@
509 \else
510 \@lockR \z@

```

```

511     \fi
512 \fi}
513
514 \renewcommand*{\do@lockoff}{%
515     \ifledRcol
516         \do@lockoffR
517     \else
518         \do@lockoffL
519     \fi}
520 \global\let\lock@off=\do@lockoff
521

```

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

```

522 \providecommand*{\n@num}{%
523 \renewcommand*{\n@num}{%
524     \ifledRcol
525         \xright@appenditem{\the\absline@numR}\to\actionlines@listR
526         \xright@appenditem{-1007}\to\actions@listR
527     \else
528         \n@num@reg
529     \fi}
530

```

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

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

```

531     \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.

```

532 \renewcommand*{\@ref}[2]{%
533     \ifledRcol
534         \global\insert@countR=#1\relax
535         \loop\ifnum\insert@countR>\z@
536             \xright@appenditem{\the\absline@numR}\to\insertlines@listR
537             \global\advance\insert@countR \m@ne
538         \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.

```

539 \begingroup
540   \let\@ref=\dummy@ref
541   \let\page@action=\relax
542   \let\sub@action=\relax
543   \let\set@line@action=\relax
544   \let\@lab=\relax
545   #2
546   \global\endpage@num=\page@numR
547   \global\endline@num=\line@numR
548   \global\endsubline@num=\subline@numR
549 \endgroup

```

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

```

550   \xright@appenditem%
551   {\the\page@numR|\the\line@numR|}%
552   \ifsublines@ \the\subline@numR \else 0\fi}%
553   \the\endpage@num|\the\endline@num|}%
554   \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.

```

555   #2
556   \else

```

And when not in right text

```

557   \@ref@reg{#1}{#2}%
558   \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.

```

559 \providecommand*\@pend}[1]{}
560 \renewcommand*\@pend}[1]{}%
561   \ifbypstart@\global\line@num=0\fi%
562   \xright@appenditem{#1}\to\linesinpar@listL}
563 \providecommand*\@pendR}[1]{}
564 \renewcommand*\@pendR}[1]{}%
565   \ifbypstart@R\global\line@numR=0\fi
566   \xright@appenditem{#1}\to\linesinpar@listR}
567

```

`\@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.

```

568 \providecommand*\@lopL}[1]{}

```

```

569 \renewcommand*{\@lopL}[1]{%
570   \xright@appenditem{#1}\to\linesonpage@listL}
571 \providecommand*{\@lopR}[1]{%
572   \renewcommand*{\@lopR}[1]{%
573     \xright@appenditem{#1}\to\linesonpage@listR}
574

```

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`.

```

575 \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
576 \newif\iffirst@linenum@out@R
577   \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.

```

578 \newcommand*{\line@list@stuffR}[1]{%
579   \read@linelist{#1}%
580   \iffirst@linenum@out@R
581     \immediate\closeout\linenum@outR
582     \global\first@linenum@out@Rfalse
583     \immediate\openout\linenum@outR=#1
584   \else
585     \if@minipage%
586       \if@ledgroup%
587         \closeout\linenum@outR
588         \openout\linenum@outR=#1
589       \else
590         \immediate\closeout\linenum@outR
591         \immediate\openout\linenum@outR=#1\relax
592       \fi
593     \else
594       \closeout\linenum@outR%
595       \openout\linenum@outR=#1\relax%
596     \fi
597   \fi}
598

```

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

```

599 \newcommand*{\new@lineR}{%
600   \write\linenum@outR{\string\@l[\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.

601 \renewcommand*{\flag@start}{%
602   \ifledRcol
603     \edef\next{\write\linenum@outR{%
604       \string\@ref[\the\insert@countR] []}%
605     \next
606   \else
607     \edef\next{\write\linenum@out{%
608       \string\@ref[\the\insert@count] []}%
609     \next
610   \fi}
611 \renewcommand*{\flag@end}{%
612   \ifledRcol
613     \write\linenum@outR{[]}%
614   \else
615     \write\linenum@out{[]}%
616   \fi}

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

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

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

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

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

634 \renewcommand*{\setline}[1]{%
635   \ifnum#1<\z@

```

```

636     \led@warn@BadSetline
637 \else
638     \ifledRcol \write\linenum@outR{\string\@set[#1]}%
639     \else      \write\linenum@out{\string\@set[#1]}%
640     \fi
641 \fi}

```

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

```

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

```

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

```

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

```

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

```

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

```

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.

```
666 \long\def\critext#1#2/{\leavevmode
667   \begingroup
668     \renewcommand{\@tag}{\no@expands #1}%
669     \set@line
670     \ifl@rcol \global\insert@countR \z@
671     \else      \global\insert@count \z@ \fi
672     \ignorespaces #2\relax
673     \@ifundefined{xpg@main@language}{%if not polyglossia
674       \flag@start}%
675       {\if@RTL\flag@end\else\flag@start\fi% be careful on the direction of writing with p
676       }%
677   \endgroup
678   \showlemma{#1}%
679   \ifx\end@lemmas\empty \else
680     \gl@p\end@lemmas\to\x@lemma
681     \x@lemma
682     \global\let\x@lemma=\relax
683   \fi
684   \@ifundefined{xpg@main@language}{%if not polyglossia
685     \flag@end}%
686     {\if@RTL\flag@start\else\flag@end\fi% be careful on the direction of writing with p
687     }
688 }
```

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

```
689 \renewcommand{\edtext}[2]{\leavevmode
690   \begingroup%
691     \renewcommand{\@tag}{\no@expands #1}%
692     \set@line%
693     \ifl@rcol \global\insert@countR \z@%
694     \else      \global\insert@count \z@ \fi%
695     \ignorespaces #2\relax%
696     \@ifundefined{xpg@main@language}{%if not polyglossia
697       \flag@start}%
698       {\if@RTL\flag@end\else\flag@start\fi% be careful on the direction of writing with p
```



```

699     }%
700   \endgroup%
701   \showlemma{#1}%
702   \ifx\end@lemmas\empty \else%
703     \glp\end@lemmas\to\x@lemma%
704     \x@lemma%
705     \global\let\x@lemma=\relax%
706   \fi%
707   \@ifundefined{xpg@main@language}{%if not polyglossia
708     \flag@end}%
709     {\ifRTL\flag@start\else\flag@end\fi% be careful on the direction of writing with polyglossia
710     }%
711 }
712

```

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

```

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

```

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 734 \newenvironment{pairs}{%}
735   \l@d@pairingtrue
736   \l@d@pagingfalse

```

```

737 }{%
738   \l@dpairingfalse
739 }

```

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

```

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

```

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

```

757 \newif\ifinstanzaL
758 \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.

```

759 \newenvironment{Leftside}{%
760   \ledRcolfalse
761   \let\beginnumbering\beginnumbering\setcounter{pstartL}{1}
762   \let\pstart\pstartL
763   \let\thepstart\thepstartL
764   \let\pend\pendL
765   \let\memorydump\memorydumpL
766   \Leftsidehook
767   \let\oldstanza\stanza
768   \renewcommand{\stanza}{\oldstanza\global\instanzaLtrue}
769 }{
770   \let\stanza\oldstanza
771   \Leftsidehookend}

```

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

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

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

```

777 \newenvironment{Rightside}{%
778   \ledRcoltrue
779   \let\beginnumbering\beginnumberingR
780   \let\endnumbering\endnumberingR
781   \let\pausenumbering\pausenumberingR
782   \let\resumenumbering\resumenumberingR
783   \let\memorydump\memorydumpR
784   \let\thepstart\thepstartR
785   \let\pstart\pstartR
786   \let\pend\pendR
787   \let\lineation\lineationR
788   \Rightsidehook
789   \let\oldstanza\stanza
790   \renewcommand{\stanza}{\oldstanza\global\instanzaRtrue}
791 }{%
792   \ledRcolfalse
793   \let\stanza\oldstanza
794   \Rightsidehookend
795 }
796
```

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
`\one@lineR` decomposition.
`\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.

```
797 \newcount\num@linesR
798 \newbox\one@lineR
799 \newcount\par@lineR
```

`\pstartL` changesv1.12012/09/25Add `\labelpstarttrue` (from `eledmac`). changesv1.1.12012/10/01Correct
`\pstartR` `\pstartR` bug introduced by 1.1. `\pstart` starts the paragraph by clearing the `\inserts@list` list and other relevant variables, and then arranges for the subsequent text to go into the appropriate box. `\pstart` needs to appear at the start of every paragraph that's to be numbered.

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

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

```
800
801 \newcounter{pstartL}
802 \newcounter{pstartLold}
803 \renewcommand{\thepstartL}{\bfseries\@arabic\c@pstartL}. }
804 \newcounter{pstartR}
805 \newcounter{pstartRold}
806 \renewcommand{\thepstartR}{\bfseries\@arabic\c@pstartR}. }
807
808 \newcommand*{\pstartL}{
809   \if@nobreak
810     \let\@oldnobreak\@nobreaktrue
811   \else
812     \let\@oldnobreak\@nobreakfalse
813   \fi
814   \@nobreaktrue
815   \ifnumbering \else
816     \led@err@PstartNotNumbered
817     \beginnumbering
818   \fi
819   \ifnumberedpar@
820     \led@err@PstartInPstart
821   \pend
822   \fi
```

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

```
823 \ifpstart@rtedL\else
```

```

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

```

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

841 \newcommand*{\pstartR}{
842 \if@nobreak
843   \let\@oldnobreak\@nobreaktrue
844 \else
845   \let\@oldnobreak\@nobreakfalse
846 \fi
847   \@nobreaktrue
848 \ifnumberingR \else
849   \led@err@PstartNotNumbered
850   \beginnumberingR
851 \fi
852 \ifnumberedpar@
853   \led@err@PstartInPstart
854   \pendR
855 \fi
856 \ifpst@rtedR\else
857   \setcounter{pstartRold}{\value{pstartR}}%
858   \list@clear{\inserts@listR}%
859   \global\let\next@insertR=\empty
860   \global\pst@rtedRtrue
861 \fi
862 \begingroup\normal@pars
863 \global\advance\l@dnumpstartsR \@ne
864 \ifnum\l@dnumpstartsR>\l@dc@maxchunks
865   \led@err@TooManyPstarts
866   \global\l@dnumpstartsR=\l@dc@maxchunks
867 \fi
868 \global\setnamebox{\l@dRcolrawbox\the\l@dnumpstartsR}=\vbox\bgroup\ifautopar\else\ifnumberpstart\ifs
869   \hsize=\Rcolwidth
870 \numberedpar@true

```

```

871 \iflabelpstart\protected@edef\@currentlabel
872     {\p@pstartR\thepstartR}\fi
873 }

```

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

```

874 \newcommand*{\pendL}{\ifnumbering \else
875     \led@err@PendNotNumbered
876 \fi
877 \ifnumberedpar@ \else
878     \led@err@PendNoPstart
879 \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.

```

880 \l@dzeropenalties
881 \endgraf\global\num@lines=\prevgraf\egroup
882 \global\par@line=0
883 \endgroup
884 \ignorespaces
885 \@oldnbreak
886 \ifnumberpstart
887 \addtocounter{pstartL}{1}
888 \fi}
889

```

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

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

```

890 \newcommand*{\pendR}{\ifnumberingR \else
891     \led@err@PendNotNumbered
892 \fi
893 \ifnumberedpar@ \else
894     \led@err@PendNoPstart
895 \fi
896 \l@dzeropenalties
897 \endgraf\global\num@linesR=\prevgraf\egroup
898 \global\par@lineR=0
899 \endgroup
900 \ignorespaces
901 \@oldnbreak
902 \ifnumberpstart
903 \addtocounter{pstartR}{1}
904 \fi}
905 }
906

```

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.

```
907 \newbox\l@dleftbox
908 \newbox\l@drightbox
909
```

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

```
\countRline 910 \newcount\countLline
911 \countLline \z@
912 \newcount\countRline
913 \countRline \z@
914
```

`\@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 915 \newcount\@donereallinesL
916 \newcount\@donetotallinesL
917 \newcount\@donereallinesR
918 \newcount\@donetotallinesR
919
```

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

```
920 \newcommand*{\do@lineL}{%
921 \advance\countLline \@ne
922 \ifvbox\namebox{\l@dLcolrawbox\the\l@dpscl}%
923 {\vbadness=10000
924 \splittopskip=\z@
925 \do@lineLhook
926 \l@demptyd@ta
927 \global\setbox\one@line=\vsplit\namebox{\l@dLcolrawbox\the\l@dpscl}
928 to\baselineskip}%
929 \unvbox\one@line \global\setbox\one@line=\lastbox
930 \getline@numL
931 \ifnum\@lock>\@ne\inserthangingsymboltrue\else\inserthangingsymbolfalse\fi
932 \setbox\l@dleftbox
933 \hb@xt@ \l@colwidth{%
934 \affixpstart@numL
935 \affixline@num
936 \l@dld@ta
937 \add@inserts
938 \affixside@note
```

```

939 \l@dlsn@te
940 {\ledllfill\hb@xt@ \wd\one@line{\do@insidelineLhook\inserthangingsymbolL\new@line\l@d
941 \l@drsn@te
942 }}%
943 \add@penaltiesL
944 \global\advance\@donereallinesL\@ne
945 \global\advance\@donetotallinesL\@ne
946 \else
947 \setbox\l@dleftbox \hb@xt@ \Lcolwidth{\hspace*\Lcolwidth}}%
948 \global\advance\@donetotallinesL\@ne
949 \fi}
950
951

```

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

```

\do@lineRhook 952 \newcommand*{\do@lineLhook}{}
\do@insidelineLhook 953 \newcommand*{\do@lineRhook}{}
\do@insidelineRhook 954 \newcommand*{\do@insidelineLhook}{}
955 \newcommand*{\do@insidelineRhook}{}
956

```

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

```

957 \newcommand*{\do@lineR}{%
958 \advance\countRline \@ne
959 \ifvbox\namebox{\l@dRcolrawbox\the\l@dpscR}%
960 {\vbadness=10000
961 \splittopskip=\z@
962 \do@lineRhook
963 \l@emptyd@ta
964 \global\setbox\one@lineR=\vsplit\namebox{\l@dRcolrawbox\the\l@dpscR}
965 to\baselineskip}%
966 \unvbox\one@lineR \global\setbox\one@lineR=\lastbox
967 \getline@numR
968 \ifnum\@lockR>\@ne\inserthangingsymbolRtrue\else\inserthangingsymbolRfalse\fi
969 \setbox\l@drightbox
970 \hb@xt@ \Rcolwidth{%
971 \affixstart@numR
972 \affixline@numR
973 \l@dld@ta
974 \add@insertsR
975 \affixside@noteR
976 \l@dlsn@te
977 {\correcthangingR\ledllfill\hb@xt@ \wd\one@lineR{\do@insidelineRhook\inserthangingsymb
978 \l@drsn@te
979 }}%
980 \add@penaltiesR
981 \global\advance\@donereallinesR\@ne
982 \global\advance\@donetotallinesR\@ne

```



```

983 \else
984   \setbox\l@drightbox \hb@xt@ \Rcolwidth{\hspace*{\Rcolwidth}}
985   \global\advance\@donetotallinesR\@ne
986 \fi}
987
988

```

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.

```

989 \newcommand*{\getline@numR}{%
990   \global\advance\absline@numR \@ne
991   \do@actionsR
992   \do@ballastR
993   \ifnumberline
994     \ifsublines@
995       \ifnum\sub@lockR<\tw@
996         \global\advance\subline@numR \@ne
997       \fi
998     \else
999       \ifnum\@lockR<\tw@
1000         \global\advance\line@numR \@ne
1001         \global\subline@numR \z@
1002       \fi
1003     \fi
1004 \fi
1005 }
1006 \newcommand*{\getline@numL}{%
1007   \global\advance\absline@num \@ne
1008   \do@actions
1009   \do@ballast
1010   \ifnumberline
1011     \ifsublines@
1012       \ifnum\sub@lock<\tw@
1013         \global\advance\subline@num \@ne
1014       \fi
1015     \else
1016       \ifnum\@lock<\tw@
1017         \global\advance\line@num \@ne
1018         \global\subline@num \z@
1019       \fi
1020     \fi
1021 \fi
1022 }
1023
1024

```

`\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.

```

1025 \newcommand*{\do@ballastR}{\global\ballast@count=\z@
1026 \begingroup
1027   \advance\absline@numR \@ne
1028   \ifnum\next@actionlineR=\absline@numR
1029     \ifnum\next@actionR>-1001
1030       \global\advance\ballast@count by -\c@ballast
1031     \fi
1032   \fi
1033 \endgroup}

```

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

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

```

1034 \newcommand*{\do@actions@fixedcodeR}{%
1035   \ifcase\@l@dtmpcnta%
1036     \or% % 1001
1037       \global\sublines@true
1038     \or% % 1002
1039       \global\sublines@false
1040     \or% % 1003
1041       \global\@lockR=\@ne
1042     \or% % 1004
1043       \ifnum\@lockR=\tw@
1044         \global\@lockR=\thr@@
1045       \else
1046         \global\@lockR=\z@
1047       \fi
1048     \or% % 1005
1049       \global\sub@lockR=\@ne
1050     \or% % 1006
1051       \ifnum\sub@lockR=\tw@
1052         \global\sub@lockR=\thr@@
1053       \else
1054         \global\sub@lockR=\z@
1055       \fi
1056     \or% % 1007
1057       \l@dskipnumbertrue
1058     \else
1059       \led@warn@BadAction
1060     \fi}
1061
1062
1063 \newcommand*{\do@actionsR}{%
1064   \global\let\do@actions@nextR=\relax
1065   \@l@dtmpcntb=\absline@numR
1066   \ifnum\@l@dtmpcntb<\next@actionlineR\else

```

```

1067 \ifnum\next@actionR>-1001\relax
1068 \global\page@numR=\next@actionR
1069 \ifbypage@R
1070 \global\line@numR \z@ \global\subline@numR \z@
1071 \fi
1072 \else
1073 \ifnum\next@actionR<-4999\relax % 9/05 added relax here
1074 \@l@dttempcnta=-\next@actionR
1075 \advance\@l@dttempcnta by -5001\relax
1076 \ifsublines@
1077 \global\subline@numR=\@l@dttempcnta
1078 \else
1079 \global\line@numR=\@l@dttempcnta
1080 \fi
1081 \else
1082 \@l@dttempcnta=-\next@actionR
1083 \advance\@l@dttempcnta by -1000\relax
1084 \do@actions@fixedcodeR
1085 \fi
1086 \fi
1087 \ifx\actionlines@listR\empty
1088 \gdef\next@actionlineR{1000000}%
1089 \else
1090 \gl@p\actionlines@listR\to\next@actionlineR
1091 \gl@p\actions@listR\to\next@actionR
1092 \global\let\do@actions@nextR=\do@actionsR
1093 \fi
1094 \fi
1095 \do@actions@nextR}
1096

```

14.4 Line number printing

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

```

\ch@cksub@l@ckR 1097
\ch@ck@l@ckR 1098 \providecommand*{\l@dcalcnun}[3]{%
\fx@l@ckR 1099 \ifnum #1 > #2\relax
\affixline@numR 1100 \@l@dttempcnta = #1\relax
1101 \advance\@l@dttempcnta by -#2\relax
1102 \divide\@l@dttempcnta by #3\relax
1103 \multiply\@l@dttempcnta by #3\relax
1104 \advance\@l@dttempcnta by #2\relax
1105 \else
1106 \@l@dttempcnta=#2\relax
1107 \fi}
1108
1109 \newcommand*{\ch@cksub@l@ckR}{%
1110 \ifcase\sub@lockR
1111 \or

```

```

1112 \ifnum\sublock@disp=\@ne
1113 \@l@tempcntb \z@ \@l@tempcnta \@ne
1114 \fi
1115 \or
1116 \ifnum\sublock@disp=\tw@
1117 \else
1118 \@l@tempcntb \z@ \@l@tempcnta \@ne
1119 \fi
1120 \or
1121 \ifnum\sublock@disp=\z@
1122 \@l@tempcntb \z@ \@l@tempcnta \@ne
1123 \fi
1124 \fi}
1125
1126 \newcommand*{\ch@ck@l@ckR}{%
1127 \ifcase\@lockR
1128 \or
1129 \ifnum\lock@disp=\@ne
1130 \@l@tempcntb \z@ \@l@tempcnta \@ne
1131 \fi
1132 \or
1133 \ifnum\lock@disp=\tw@
1134 \else
1135 \@l@tempcntb \z@ \@l@tempcnta \@ne
1136 \fi
1137 \or
1138 \ifnum\lock@disp=\z@
1139 \@l@tempcntb \z@ \@l@tempcnta \@ne
1140 \fi
1141 \fi}
1142
1143 \newcommand*{\f@x@l@cksR}{%
1144 \ifcase\@lockR
1145 \or
1146 \global\@lockR \tw@
1147 \or \or
1148 \global\@lockR \z@
1149 \fi
1150 \ifcase\sub@lockR
1151 \or
1152 \global\sub@lockR \tw@
1153 \or \or
1154 \global\sub@lockR \z@
1155 \fi}
1156
1157
1158 \newcommand*{\affixline@numR}{%
1159 \ifnumberline
1160 \ifl@dskipnumber
1161 \global\l@dskipnumberfalse

```

```

1162 \else
1163   \ifsublines@
1164     \@l@tempcntb=\subline@numR
1165     \l@dcalcnum{\subline@numR}{\c@firstsublinenumR}{\c@sublinenumincrementR}%
1166     \ch@cksub@lockR
1167   \else
1168     \@l@tempcntb=\line@numR
1169     \ifx\linenumberlist\empty
1170       \l@dcalcnum{\line@numR}{\c@firstlinenumR}{\c@linenumincrementR}%
1171     \else
1172       \@l@tempcnta=\line@numR
1173       \edef\rem@inder{\linenumberlist,\number\line@numR,}%
1174       \edef\sc@n@list{\def\noexpand\sc@n@list
1175         ###1,\number\@l@tempcnta,###2|{\def\noexpand\rem@inder{###2}}}%
1176       \sc@n@list\expandafter\sc@n@list\rem@inder|
1177       \ifx\rem@inder\empty\advance\@l@tempcnta\@ne\fi
1178     \fi
1179     \ch@ck@l@ckR
1180   \fi
1181   \ifnum\@l@tempcnta=\@l@tempcntb
1182     \if@twocolumn
1183       \if@firstcolumn
1184         \gdef\l@dld@ta{\llap{\leftlinenumR}}}%
1185       \else
1186         \gdef\l@drd@ta{\rlap{\rightlinenumR}}}%
1187     \fi
1188   \else
1189     \@l@tempcntb=\line@marginR
1190     \ifnum\@l@tempcntb>\@ne
1191       \advance\@l@tempcntb by\page@numR
1192     \fi
1193     \ifodd\@l@tempcntb
1194       \gdef\l@drd@ta{\rlap{\rightlinenumR}}}%
1195     \else
1196       \gdef\l@dld@ta{\llap{\leftlinenumR}}}%
1197     \fi
1198   \fi
1199 \fi
1200 \f@x@l@ckR
1201 \fi
1202 \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 begining of this command.

```

\affixpstart@numL
\affixpstart@numR 1203
  \leftpstartnumR 1204 \newcommand*{\affixpstart@numL}{%
\rightpstartnumR 1205 \ifsidepstartnum
  \leftpstartnumL 1206 \if@twocolumn
\rightpstartnumL 1207   \if@firstcolumn
  \ifpstartnumR 1208     \gdef\l@dld@ta{\llap{\leftpstartnumL}}}%
1209     \else
1210     \gdef\l@drd@ta{\rlap{\rightpstartnumL}}}%
1211     \fi
1212   \else
1213     \@l@tempcntb=\line@margin
1214     \ifnum\@l@tempcntb>\@ne
1215       \advance\@l@tempcntb \page@num
1216     \fi
1217     \ifodd\@l@tempcntb
1218       \gdef\l@drd@ta{\rlap{\rightpstartnumL}}}%
1219     \else
1220       \gdef\l@dld@ta{\llap{\leftpstartnumL}}}%
1221     \fi
1222   \fi
1223 \fi
1224 }
1225 \newcommand*{\affixpstart@numR}{%
1226 \ifsidepstartnum
1227 \if@twocolumn
1228   \if@firstcolumn
1229     \gdef\l@dld@ta{\llap{\leftpstartnumR}}}%
1230   \else
1231     \gdef\l@drd@ta{\rlap{\rightpstartnumR}}}%
1232   \fi
1233   \else
1234     \@l@tempcntb=\line@marginR
1235     \ifnum\@l@tempcntb>\@ne
1236       \advance\@l@tempcntb \page@numR
1237     \fi
1238     \ifodd\@l@tempcntb
1239       \gdef\l@drd@ta{\rlap{\rightpstartnumR}}}%
1240     \else
1241       \gdef\l@dld@ta{\llap{\leftpstartnumR}}}%
1242     \fi
1243   \fi
1244 \fi
1245 }
1246
1247 \newcommand*{\leftpstartnumL}{
1248 \ifpstartnum
1249 \thepstartL

```

```

1250 \kern\linenumsep\global\pstartnumfalse\fi
1251 }
1252 \newcommand*{\rightpstartnumL}{
1253 \ifpstartnum\kern\linenumsep
1254 \thepstartL
1255 \global\pstartnumfalse\fi
1256 }
1257 \newif\ifpstartnumR
1258 \pstartnumRtrue
1259 \newcommand*{\leftpstartnumR}{
1260 \ifpstartnumR
1261 \thepstartR
1262 \kern\linenumsep\global\pstartnumRfalse\fi
1263 }
1264 \newcommand*{\rightpstartnumR}{
1265 \ifpstartnumR\kern\linenumsep
1266 \thepstartR
1267 \global\pstartnumRfalse\fi
1268 }

```

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.

```

1269 \list@create{\inserts@listR}

```

`\add@insertsR` The right text version.

```

\add@inserts@nextR 1270 \newcommand*{\add@insertsR}{%
1271 \global\let\add@inserts@nextR=\relax
1272 \ifx\inserts@listR\empty \else
1273 \ifx\next@insertR\empty
1274 \ifx\insertlines@listR\empty
1275 \global\noteschanged@true
1276 \gdef\next@insertR{100000}%
1277 \else
1278 \gl@p\insertlines@listR\to\next@insertR
1279 \fi
1280 \fi
1281 \ifnum\next@insertR=\absline@numR
1282 \gl@p\inserts@listR\to\@insertR
1283 \@insertR
1284 \global\let\@insertR=\undefined
1285 \global\let\next@insertR=\empty
1286 \global\let\add@inserts@nextR=\add@insertsR
1287 \fi
1288 \fi
1289 \add@inserts@nextR}
1290

```

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@tempcnta` is used to calculate and accumulate the penalty; it is initially set to the value of `\ballast@count`, which has been worked out in `\do@ballast`. Finally, the penalty is checked to see that it doesn't go below -10000 .

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

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

```
1291 \newcommand*{\add@penaltiesL}{\fi}
1292 \newcommand*{\add@penaltiesR}{\fi}
1293
```


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.

```

1294 \newcommand*{\flush@notesR}{%
1295   \xloop
1296     \ifx\inserts@listR\empty \else
1297       \glp\inserts@listR\to\@insertR
1298       \@insertR
1299       \global\let\@insertR=\undefined
1300   \repeat}
1301
```

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:

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

```

1302 \def\printlinesR#1|#2|#3|#4|#5|#6|#7|{\begingroup
1303   \setprintlines{#1}{#2}{#3}{#4}{#5}{#6}%
1304   \ifl@d@pnum #1\fullstop\fi
1305   \ifledplinenum \linenumr@p{#2}\Rlineflag\else \symplinenum\fi
1306   \ifl@d@ssub \fullstop \sublinenumr@p{#3}\fi
1307   \ifl@d@dash \endashchar\fi
1308   \ifl@d@pnum #4\fullstop\fi
1309   \ifl@d@elin \linenumr@p{#5}\Rlineflag\fi
1310   \ifl@d@esl \ifl@d@elin \fullstop\fi \sublinenumr@p{#6}\fi
1311 \endgroup}
1312
1313 \let\ledsavedprintlines\printlines
1314
```

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.

```

1315 \list@create{\labelref@listR}
```

1316

`\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.

```

1317 \renewcommand*{\edlabel}[1]{\@bsphack
1318   \ifledRcol
1319     \write\linenum@outR{\string\@lab}%
1320     \ifx\labelref@listR\empty
1321       \xdef\label@refs{\zz@@}%
1322     \else
1323       \glp\labelref@listR\to\label@refs
1324     \fi
1325     \ifvmode
1326       \advancelabel@refs
1327     \fi
1328     \protected@write\@auxout{%
1329       {\string\l@dmake@labelsR\space\thepage|\label@refs|{#1}}%
1330   \else
1331     \write\linenum@out{\string\@lab}%
1332     \ifx\labelref@list\empty
1333       \xdef\label@refs{\zz@@}%
1334     \else
1335       \glp\labelref@list\to\label@refs
1336     \fi
1337     \ifvmode
1338       \advancelabel@refs
1339     \fi
1340     \protected@write\@auxout{%
1341       {\string\l@dmake@labelsR\space\thepage|\label@refs|{#1}}%
1342     \fi
1343   \@esphack}
1344
```

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

```

1345 \def\l@dmake@labelsR#1|#2|#3|#4{%
1346   \expandafter\ifx\csname the@label#4\endcsname \relax\else
1347     \led@warn@DuplicateLabel{#4}%
1348   \fi
1349   \expandafter\gdef\csname the@label#4\endcsname{#1|#2\Rlineflag|#3}%
1350   \ignorespaces}
1351 \AtBeginDocument{%
1352   \def\l@dmake@labelsR#1|#2|#3|#4{%
1353   }
1354
```

`\@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.

```

1355 \renewcommand*{\@lab}{%
1356   \ifledRcol
1357     \xright@appenditem{\linenumr@p{\line@numR}}{|%
1358       \ifsublines@ \sublinenumr@p{\subline@numR}\else 0\fi}%
1359     \to\labelref@listR
1360   \else
1361     \xright@appenditem{\linenumr@p{\line@num}}{|%
1362       \ifsublines@ \sublinenumr@p{\subline@num}\else 0\fi}%
1363     \to\labelref@list
1364   \fi}
1365
```

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 1366 \newcount\sidenote@marginR
1367 \renewcommand*{\sidenotemargin}[1]{%
1368   \l@dgetsidenote@margin{#1}%
1369   \ifnum\@l@dttempcntb>\m@ne
1370     \ifledRcol
1371       \global\sidenote@marginR=\@l@dttempcntb
1372     \else
1373       \global\sidenote@margin=\@l@dttempcntb
1374     \fi
1375   \fi}}
1376 \sidenotemargin{right}
1377 \global\sidenote@margin=\@ne
1378
```

`\l@dlsnote` The ‘footnotes’ for left, right, and moveable sidenotes. The whole scheme is rem-

`\l@drsnote` iniscent of the critical footnotes code.

```

\l@dcsnote 1379 \renewcommand*{\l@dlsnote}[1]{%
1380   \begingroup%
1381   \newcommand{\content}{#1}%
1382   \ifnumberedpar@
1383     \ifledRcol%
1384       \xright@appenditem{\noexpand\l@dlsnote{\csexpandonce{content}}}%
1385       \to\inserts@listR
1386       \global\advance\insert@countR \@ne%
1387     \else%
1388       \xright@appenditem{\noexpand\l@dlsnote{\csexpandonce{content}}}%
1389       \to\inserts@list
1390       \global\advance\insert@count \@ne%

```

```

1391 \fi
1392 \fi\ignorespaces\endgroup}
1393 \renewcommand*{\l@drsnote}[1]{%
1394 \begingroup%
1395 \newcommand{\content}{#1}%
1396 \ifnumberedpar@
1397 \ifledRcol%
1398 \xright@appenditem{\noexpand\vl@drsnote{\csexpandonce{content}}}%
1399 \to\inserts@listR
1400 \global\advance\insert@countR \@ne%
1401 \else%
1402 \xright@appenditem{\noexpand\vl@drsnote{\csexpandonce{content}}}%
1403 \to\inserts@list
1404 \global\advance\insert@count \@ne%
1405 \fi
1406 \fi\ignorespaces\endgroup}
1407 \renewcommand*{\l@dcnote}[1]{%
1408 \begingroup%
1409 \newcommand{\content}{#1}%
1410 \ifnumberedpar@
1411 \ifledRcol%
1412 \xright@appenditem{\noexpand\vl@dcnote{\csexpandonce{content}}}%
1413 \to\inserts@listR
1414 \global\advance\insert@countR \@ne%
1415 \else%
1416 \xright@appenditem{\noexpand\vl@dcnote{\csexpandonce{content}}}%
1417 \to\inserts@list
1418 \global\advance\insert@count \@ne%
1419 \fi
1420 \fi\ignorespaces\endgroup}
1421

```

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

```

1422 \newcommand*{\affixside@noteR}{%
1423 \def\sidenotecontent@{}%
1424 \numdef{\itemcount@}{0}%
1425 \def\do##1{%
1426 \ifnumequal{\itemcount@}{0}%
1427 {%
1428 \appto\sidenotecontent@{##1}}% Not print not separator before the 1st note
1429 {\appto\sidenotecontent@{\sidenotesep ##1}%
1430 }%
1431 \numdef{\itemcount@}{\itemcount@+1}%
1432 }%
1433 \dolistloop{\l@dcnotetext}%
1434 \ifnumgreater{\itemcount@}{1}{\eledmac@warning{\itemcount@\space sidenotes on line \tl
1435 \gdef\@templ@d{}}%
1436 \ifx\@templ@d\l@dcnotetext \else%
1437 \if@twocolumn%
1438 \if@firstcolumn%

```

```

1439     \setl@dlp@rbox{\sidenotecontent@}%
1440     \else%
1441     \setl@drp@rbox{\sidenotecontent@}%
1442     \fi%
1443 \else%
1444     \l@dttempcntb=\sidenote@marginR%
1445     \ifnum\l@dttempcntb>\@ne%
1446     \advance\l@dttempcntb by\page@num%
1447     \fi%
1448     \ifodd\l@dttempcntb%
1449     \setl@drp@rbox{\sidenotecontent@t}%
1450     \else%
1451     \setl@dlp@rbox{\sidenotecontent@}%
1452     \fi%
1453 \fi%
1454 \fi}
1455

```

18 Familiar footnotes

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

```

1456 \renewcommand{\l@dbfnote}[1]{%
1457   \ifnumberedpar@
1458   \gdef\@tag{#1}%
1459   \ifledRcol%
1460     \xright@appenditem{\noexpand\vl@dbfnote{\csexpandonce{\@tag}}{\@thefnmark}}%
1461     \to\inserts@listR
1462     \global\advance\insert@countR \@ne%
1463   \else%
1464     \xright@appenditem{\noexpand\l@dbfnote{\csexpandonce{\@tag}}{\@thefnmark}}%
1465     \to\inserts@list
1466     \global\advance\insert@count \@ne%
1467   \fi
1468 \fi\ignorespaces}
1469

```

`\normalbfnoteX`

```

1470 \renewcommand{\normalbfnoteX}[2]{%
1471   \ifnumberedpar@
1472   \ifledRcol%
1473   \ifluatex
1474     \footnotelang@lua[R]%
1475   \fi
1476   \@ifundefined{xpg@main@language}%if polyglossia
1477 {}%
1478 {\footnotelang@poly[R]}%
1479   \protected@csxdef{thisfootnote}{\csuse{thefootnote#1}}%

```

```

1480 \xright@appenditem{\noexpand\vbfnoteX{#1}{#2}{\csexpandonce{thisfootnote}}}%
1481 \to\inserts@listR
1482 \global\advance\insert@countR \@ne%
1483 \else%
1484 \ifluatex
1485 \footnotelang@lua%
1486 \fi
1487 \@ifundefined{xpg@main@language}%if polyglossia
1488 {}%
1489 {\footnotelang@poly}%
1490 \protected@csxdef{thisfootnote}{\csuse{thefootnote#1}}%
1491 \xright@appenditem{\noexpand\vbfnoteX{#1}{#2}{\csexpandonce{thisfootnote}}}%
1492 \to\inserts@list
1493 \global\advance\insert@count \@ne%
1494 \fi
1495 \fi\ignorespaces}
1496

```

19 Verse

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

```

\inserthangingsymbolL
\inserthangingsymbolR 1497 \newif\ifinserthangingsymbolR
1498 \newcommand{\inserthangingsymbolL}{%
1499 \ifinserthangingsymbol%
1500 \ifinstanzaL%
1501 \hfill\hangingsymbol%
1502 \fi%
1503 \fi}
1504 \newcommand{\inserthangingsymbolR}{%
1505 \ifinserthangingsymbolR%
1506 \ifinstanzaR%
1507 \hfill\hangingsymbol%
1508 \fi%
1509 \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 1510 \newcommand{\correcthangingL}{%
1511 \ifl@dpage\else%
1512 \ifinstanzaL%
1513 \ifinserthangingsymbol%
1514 \hskip \@ifundefined{sza@00}{0}{\expandafter%

```

```

1515          \noexpand\csname sza@0@\endcsname\stanzaindentbase%
1516      \fi%
1517  \fi%
1518 \fi}
1519
1520 \newcommand{\correcthangingR}{%
1521 \ifl@dpaging\else%
1522     \ifinstanzaR%
1523         \ifinserthangingsymbolR%
1524             \hskip \@ifundefined{sza@0@}{0}{\expandafter%
1525                 \noexpand\csname sza@0@\endcsname\stanzaindentbase%
1526             \fi%
1527         \fi%
1528 \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.

```

1529 \chardef\next=\catcode'\&
1530 \catcode'\&=\active
1531

```

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

```

1532 \newenvironment{astanza}{%
1533     \startstanzahook
1534     \catcode'\&\active
1535     \global\stanza@count\@ne\stanza@modulo\@ne
1536     \ifnum\usernamecount{sza@0@}=\z@
1537         \let\stanza@hang\relax
1538         \let\endlock\relax
1539     \else
1540 %%% \interlinepenalty\@M % this screws things up, but I don't know why
1541     \rightskip\z@ plus 1fil\relax
1542     \fi
1543     \ifnum\usernamecount{szp@0@}=\z@
1544         \let\sza@penalty\relax
1545     \fi
1546     \def&{%
1547         \endlock\mbox{}%
1548         \sza@penalty
1549         \global\advance\stanza@count\@ne
1550         \@astanza@line}%
1551     \def\&{%
1552         \endlock\mbox{}
1553         \pend
1554         \endstanzaextra}%
1555     \pstart
1556     \@astanza@line

```

```

1557 }{}
1558

```

`\@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.

```

1559 \newcommand*{\@astanza@line}{%
1560   \ifnum\value{stanzaindentrepetition}=0
1561     \parindent=\csname sza@\number\stanza@count
1562             @\endcsname\stanzaindentbase
1563   \else
1564     \parindent=\csname sza@\number\stanza@modulo
1565             @\endcsname\stanzaindentbase
1566     \managestanza@modulo
1567   \fi
1568   \par
1569   \stanza@hang%\mbox{}%
1570   \ignorespaces}
1571

```

Lastly reset the modified category codes.

```

1572 \catcode'\&=\next
1573

```

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

```

\unhnamebox 1574 \providecommand*{\newnamebox}[1]{%
\unvnamebox 1575   \expandafter\newbox\csname #1\endcsname}
\namebox 1576 \providecommand*{\setnamebox}[1]{%
1577   \expandafter\setbox\csname #1\endcsname}
1578 \providecommand*{\unhnamebox}[1]{%
1579   \expandafter\unhbox\csname #1\endcsname}
1580 \providecommand*{\unvnamebox}[1]{%
1581   \expandafter\unvbox\csname #1\endcsname}
1582 \providecommand*{\namebox}[1]{%
1583   \csname #1\endcsname}
1584

```

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

```

\usenamecount 1585 \providecommand*{\newnamecount}[1]{%
1586   \expandafter\newcount\csname #1\endcsname}
1587 \providecommand*{\usenamecount}[1]{%
1588   \csname #1\endcsname}
1589

```


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
`\l@dc@maxchunks` default is 5120 chunk pairs.

```
1590 \newcount\l@dc@maxchunks
1591 \newcommand{\maxchunks}[1]{\l@dc@maxchunks=#1}
1592   \maxchunks{5120}
1593
```

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

`\l@dnumpstartsR` 1594 `\newcount\l@dnumpstartsR`
 1595

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

`\l@pscR` 1596 `\newcount\l@dpscL`
 1597 `\newcount\l@dpscR`
 1598

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

```
1599 \newcommand*{\l@dsetuprawboxes}{%
1600   \@l@tempcntb=\l@dc@maxchunks
1601   \loop\ifnum\@l@tempcntb>\z@
1602     \newnamebox{\l@dLcolrawbox\the\@l@tempcntb}
1603     \newnamebox{\l@dRcolrawbox\the\@l@tempcntb}
1604     \advance\@l@tempcntb \m@ne
1605   \repeat}
1606
```

`\l@dsetupmaxlinecounts` To be able to synchronise left and right texts we need to know the maximum num-
`\l@dzeromaxlinecounts` ber of text lines there are in each pair of chunks. `\l@dsetupmaxlinecounts` creates
`\maxchunks` new counts called `\l@dmaxlinesinpar1`, etc., and `\l@dzeromaxlinecounts`
 zeroes all of them.

```
1607 \newcommand*{\l@dsetupmaxlinecounts}{%
1608   \@l@tempcntb=\l@dc@maxchunks
1609   \loop\ifnum\@l@tempcntb>\z@
1610     \newnamecount{\l@dmaxlinesinpar\the\@l@tempcntb}
1611     \advance\@l@tempcntb \m@ne
1612   \repeat}
1613 \newcommand*{\l@dzeromaxlinecounts}{%
1614   \begingroup
1615   \@l@tempcntb=\l@dc@maxchunks
1616   \loop\ifnum\@l@tempcntb>\z@
```

```

1617 \global\usernamecount{l@dmxlinesinpar\the\@l@dttempcntb}=\z@
1618 \advance\@l@dttempcntb \m@ne
1619 \repeat
1620 \endgroup}
1621

```

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

```

1622 \AtBeginDocument{%
1623 \l@dssetuprawboxes
1624 \l@dssetupmaxlinecounts
1625 \l@dzzeromaxlinecounts
1626 \l@dnumpstartsL=\z@
1627 \l@dnumpstartsR=\z@
1628 \l@dpscL=\z@
1629 \l@dpscR=\z@}
1630

```

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 1631 \newif\ifl@dusedbabel
\l@dusedbabeltrue 1632 \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 1633 \newif\ifl@dsamelang
1634 \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.
1635 \newcommand*{\l@dchecklang}{%
1636 \l@dsamelangfalse
1637 \edef\@tempa{\theledlanguageL}\edef\@tempb{\theledlanguageR}%
1638 \ifx\@tempa\@tempb
1639 \l@dsamelangtrue

```

```

1640 \fi}
1641

```

`\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 `\`.

```

1642 \newcommand*{\l@dbbl@set@language}[1]{%
1643   \edef\language#1}%
1644   \select@language{\language}%
1645   \if@filesw
1646     \protected@write\auxout{}\string\select@language{\language}%
1647     \addtocontents{toc}{\string\select@language{\language}%
1648     \addtocontents{lof}{\string\select@language{\language}%
1649     \addtocontents{lot}{\string\select@language{\language}%
1650   \fi}
1651

```

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 1652 \providecommand{\selectlanguage}[1]{%
1653   \newcommand*{\l@duselanguage}[1]{%
1654     \gdef\theledlanguageL{
1655     \gdef\theledlanguageR{
1656

```

Now do the babel fix or polyglossia, if necessary.

```

1657 \AtBeginDocument{%
1658   \@ifundefined{xpg@main@language}{%
1659     \@ifundefined{bbl@main@language}{%

```

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

```

1660   \l@dusedbabelfalse
1661   \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.

```

1662   \l@dusedbabeltrue
1663   \let\l@doldselectlanguage\selectlanguage
1664   \let\l@doldbbl@set@language\bbl@set@language
1665   \let\bbl@set@language\l@dbbl@set@language
1666   \renewcommand{\selectlanguage}[1]{%
1667     \l@doldselectlanguage{#1}%

```

```

1668     \ifledRcol \gdef\theledlanguageR{#1}%
1669     \else      \gdef\theledlanguageL{#1}%
1670     \fi}

```

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

```

1671     \renewcommand*{\l@duselanguage}[1]{%
1672     \l@doldselectlanguage{#1}}

```

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

```

1673     \gdef\theledlanguageL{\bbl@main@language}%
1674     \gdef\theledlanguageR{\bbl@main@language}%
1675     }%
1676 }

```

If on Polyglossia

```

1677 {   \apptocmd{\xpg@set@language}{%
1678     \ifledRcol \gdef\theledlanguageR{#1}%
1679     \else      \gdef\theledlanguageL{#1}%
1680     \fi}%
1681     \let\l@duselanguage\xpg@set@language
1682     \gdef\theledlanguageL{\xpg@main@language}%
1683     \gdef\theledlanguageR{\xpg@main@language}%
1684 % \end{macrocode}
1685 % That's it.
1686 % \begin{macrocode}
1687 }}

```

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.

```

1688 \newcommand*\Columns{%
1689     \setcounter{pstartL}{\value{pstartLold}}
1690     \setcounter{pstartR}{\value{pstartRold}}
1691     \ifnum\l@dnumpstartsL=\l@dnumpstartsR\else
1692         \led@err@BadLeftRightPstarts{\the\l@dnumpstartsL}{\the\l@dnumpstartsR}%
1693     \fi

```

Start a group and zero counters, etc.

```

1694     \begingroup
1695     \l@dzeropenalties
1696     \endgraf\global\num@lines=\prevgraf
1697     \global\num@linesR=\prevgraf
1698     \global\par@line=\z@
1699     \global\par@lineR=\z@
1700     \global\l@dpscL=\z@
1701     \global\l@dpscR=\z@

```

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

```
1702     \check@pstarts
1703     \loop\if@pstarts
1704         \global\pstartnumtrue
1705         \global\pstartnumRtrue
```

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

```
1706     \global\advance\l@dpscL \@ne
1707     \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

```
1708     \checkraw@text
1709     \l@dcchecklang
1710 {     \loop\ifaraw@text
```

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

```
1711         \ifl@dsamelang
1712             \do@lineL
1713             \do@lineR
1714         \else
1715             \l@duselanguage{\theledlanguageL}%
1716             \do@lineL
1717             \l@duselanguage{\theledlanguageR}%
1718             \do@lineR
1719         \fi
1720         \hb@xt@ \hsize{%
1721             \hfill \unhbox\l@dleftbox
1722             \hfill \columnseparator \hfill
1723             \unhbox\l@drightbox
1724         }%
1725         \checkraw@text
1726     \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.

```
1727     \@writelinesinparL
1728     \@writelinesinparR
1729     \check@pstarts
1730     \ifbypstart@
1731         \write\linenum@out{\string\@set[1]}
1732         \resetprevline@
1733     \fi
1734     \ifbypstart@R
1735         \write\linenum@outR{\string\@set[1]}
1736         \resetprevline@
1737     \fi
```

```

1738     \addtocounter{pstartL}{1}
1739     \addtocounter{pstartR}{1}
1740     \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.

```

1741     \flush@notes
1742     \flush@notesR
1743     \endgroup

1744     \global\l@dpscL=\z@
1745     \global\l@dpscR=\z@
1746     \global\l@dnumpstartsL=\z@
1747     \global\l@dnumpstartsR=\z@
1748     \ignorespaces
1749     \global\instanzaLfalse
1750     \global\instanzaRfalse}
1751

```

`\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).

```

1752 \newcommand*{\columnseparator}{%
1753   \smash{\rule[-0.2\baselineskip]{\columnrulewidth}{1.05\baselineskip}}}
1754 \newdimen\columnrulewidth
1755   \columnrulewidth=\z@
1756

```

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

```

\@pstartstrue 1757 \newif\if@pstarts
\@pstartsfalse 1758 \newcommand*{\check@pstarts}{%
\check@pstarts 1759   \@pstartsfalse
1760   \ifnum\l@dnumpstartsL>\l@dpscL
1761     \@pstartstrue
1762   \else
1763     \ifnum\l@dnumpstartsR>\l@dpscR
1764       \@pstartstrue
1765     \fi
1766   \fi
1767 }
1768

```

`\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 sets `\araw@textfalse`.

```

\checkraw@text 1769 \newif\ifaraw@text
1770   \araw@textfalse
1771 \newcommand*{\checkraw@text}{%
1772   \araw@textfalse

```

```

1773 \ifvbox\namebox{l@dLcolrawbox\the\l@dpscL}
1774 \araw@texttrue
1775 \else
1776 \ifvbox\namebox{l@dRcolrawbox\the\l@dpscR}
1777 \araw@texttrue
1778 \fi
1779 \fi
1780 }
1781

```

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

```

1782 \newcommand*{\@writelinesinparL}{%
1783 \edef\next{%
1784 \write\linenum@out{\string\@pend[\the\@donereallinesL]}}%
1785 \next
1786 \global\@donereallinesL \z@}
1787 \newcommand*{\@writelinesinparR}{%
1788 \edef\next{%
1789 \write\linenum@outR{\string\@pendR[\the\@donereallinesR]}}%
1790 \next
1791 \global\@donereallinesR \z@}
1792

```

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` 1793 \newcount\numpagelinesL
1794 \newcount\numpagelinesR
1795 \newcount\l@dminpagelines
1796

`\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.

```

1797 \newcommand*{\Pages}{%
1798 \setcounter{pstartL}{\value{pstartLold}}
1799 \setcounter{pstartR}{\value{pstartRold}}
1800 \typeout{}
1801 \typeout{***** PAGES *****}
1802 \ifnum\l@dnumpstartsL=\l@dnumpstartsR\else
1803 \led@err@BadLeftRightPstarts{\the\l@dnumpstartsL}{\the\l@dnumpstartsR}%
1804 \fi

```

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

```

1805 \cleartol@evenpage
1806 \begingroup
1807 \l@dzeropenalties
1808 \endgraf\global\num@lines=\prevgraf
1809 \global\num@linesR=\prevgraf
1810 \global\par@line=\z@
1811 \global\par@lineR=\z@
1812 \global\l@dpscL=\z@
1813 \global\l@dpscR=\z@
1814 \writtenlinesLfalse
1815 \writtenlinesRfalse

```

Check if there are chunks to be processed.

```

1816 \check@pstarts
1817 \loop\if@pstarts

```

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

```

1818 \global\advance\l@dpscL \@ne
1819 \global\advance\l@dpscR \@ne

```

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

```

1820 \getlinesfromparlistL
1821 \getlinesfromparlistR
1822 \l@dcalc@maxoftwo{\@cs@linesinparL}{\@cs@linesinparR}%
1823 {\usernamecount{l@dmxlinesinpar\the\l@dpscL}}%
1824 \check@pstarts
1825 \repeat

```

Zero the counts again, ready for the next bit.

```

1826 \global\l@dpscL=\z@
1827 \global\l@dpscR=\z@

```

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

```

1828 \getlinesfrompagelistL
1829 \getlinesfrompagelistR
1830 \l@dcalc@minoftwo{\@cs@linesonpageL}{\@cs@linesonpageR}%
1831 {\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.

```

1832 \check@pstarts
1833 \if@pstarts

```

Increment the chunk counts to get the first pair.

```

1834 \global\advance\l@dpscL \@ne
1835 \global\advance\l@dpscR \@ne

```


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

```

1836      \global\@donereallinesL=\z@
1837      \global\@donetotallinesL=\z@
1838      \global\@donereallinesR=\z@
1839      \global\@donetotallinesR=\z@

```

Start a loop over the boxes (chunks).

```

1840      \checkraw@text

1841 %      \begingroup
1842 {      \loop\ifaraw@text

```

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

```

1843          \checkpageL
1844          \l@duselanguage{\theledlanguageL}%
1845 %%%          \begingroup
1846 {          \loop\ifl@dsamepage
1847

```

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

```

1848          \do@lineL
1849          \advance\numpagelinesL \@ne
1850          \ifshiftedpstarts
1851              \ifdim\ht\l@dleftbox>0pt\hb@xt@ \hsize{\ledstrutL\unhbox\l@dleftbox}\fi%
1852          \else
1853              \hb@xt@ \hsize{\ledstrutL\unhbox\l@dleftbox}%
1854          \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.

```

1855
1856          \get@nextboxL
1857          \checkpageL
1858      \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.

```

1859          \ifl@dpagfull
1860          \@writelinesonpageL{\the\numpagelinesL}%
1861      \else
1862          \@writelinesonpageL{1000}%
1863      \fi

```

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

```

1864          \numpagelinesL \z@
1865          \clearl@dleftpage }%

```

Now do the same for the right text.

```

1866      \checkpageR
1867      \l@duselanguage{\theledlanguageR}%
1868 {
1869      \loop\ifl@dsamepage
1869      \do@lineR
1870      \advance\numpagelinesR \@ne
1871      \ifshiftedpstarts
1872      \ifdim\ht\l@drightbox>0pt\hb@xt@ \hsize{\ledstrutR\unhbox\l@drightbox}%
1873      \else
1874      \hb@xt@ \hsize{\ledstrutR\unhbox\l@drightbox}%
1875      \fi
1876      \get@nextboxR
1877      \checkpageR
1878      \repeat
1879      \ifl@dpagfull
1880      \@writelinesonpageR{\the\numpagelinesR}%
1881      \else
1882      \@writelinesonpageR{1000}%
1883      \fi
1884      \numpagelinesR=\z@

```

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

```

1885      \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.

```

1886      \checkraw@text
1887      \ifaraw@text
1888      \getlinesfrompagelistL
1889      \getlinesfrompagelistR
1890      \l@dcalc@minoftwo{\@cs@linesonpageL}{\@cs@linesonpageR}%
1891      {\l@dminpagelines}%
1892      \fi
1893      \repeat}

```

We have now output the text from all the chunks.

```

1894      \fi

```

Make sure that there are no inserts hanging around.

```

1895      \flush@notes
1896      \flush@notesR
1897      \endgroup

```

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

```

1898      \global\l@dpscL=\z@
1899      \global\l@dpscR=\z@
1900      \global\l@dnpstartsL=\z@
1901      \global\l@dnpstartsR=\z@
1902      \global\instanzaLfalse

```

```

1903 \global\instanzaRfalse
1904 \ignorespaces}
1905

```

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

```

\ledstrutR 1906 \newcommand*{\ledstrutL}{\strut}
1907 \newcommand*{\ledstrutR}{\strut}
1908

```

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

```

1909 \providecommand{\cleartoevenpage}[1][\@empty]{%
1910 \clearpage
1911 \ifodd\c@page\hbox{ }#1\clearpage\fi}
1912 \newcommand*{\cleartol@devenpage}{%
1913 \ifdim\pagetotal<\topskip% on an empty page
1914 \else
1915 \clearpage
1916 \fi
1917 \ifodd\c@page\hbox{ }\clearpage\fi}
1918 \newcommand*{\clearl@dleftpage}{%
1919 \clearpage
1920 \ifodd\c@page\else
1921 \led@err@LeftOnRightPage
1922 \hbox{ }%
1923 \cleardoublepage
1924 \fi}
1925 \newcommand*{\clearl@drighthpage}{%
1926 \clearpage
1927 \ifodd\c@page
1928 \led@err@RightOnLeftPage
1929 \hbox{ }%
1930 \cleartoevenpage
1931 \fi}
1932

```

`\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 1933 \newcommand*{\getlinesfromparlistL}{%
1934 \ifx\linesinpar@listL\empty
1935 \gdef\@cs@linesinparL{0}%
1936 \else
1937 \glp\linesinpar@listL\to\@cs@linesinparL
1938 \fi}
1939 \newcommand*{\getlinesfromparlistR}{%

```

```

1940 \ifx\linesinpar@listR\empty
1941   \gdef\@cs@linesinparR{0}%
1942 \else
1943   \gl@p\linesinpar@listR\to\@cs@linesinparR
1944 \fi}
1945
\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
1946 \newcommand*\getlinesfrompagelistL{%
1947   \ifx\linesonpage@listL\empty
1948     \gdef\@cs@linesonpageL{1000}%
1949   \else
1950     \gl@p\linesonpage@listL\to\@cs@linesonpageL
1951   \fi}
1952 \newcommand*\getlinesfrompagelistR{%
1953   \ifx\linesonpage@listR\empty
1954     \gdef\@cs@linesonpageR{1000}%
1955   \else
1956     \gl@p\linesonpage@listR\to\@cs@linesonpageR
1957   \fi}
1958
\@writelinesonpageL These macros output the number of lines on a page to the section file in the form
\@writelinesonpageR of \@lopL or \@lopR macros.
1959 \newcommand*\@writelinesonpageL[1]{%
1960   \edef\next{\write\linenum@out{\string\@lopL{#1}}}%
1961   \next}
1962 \newcommand*\@writelinesonpageR[1]{%
1963   \edef\next{\write\linenum@outR{\string\@lopR{#1}}}%
1964   \next}
1965
\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>.
1966 \newcommand*\l@dcalc@maxoftwo[3]{%
1967   \ifnum #2>#1\relax
1968     #3=#2\relax
1969   \else
1970     #3=#1\relax
1971   \fi}
1972 \newcommand*\l@dcalc@minoftwo[3]{%
1973   \ifnum #2<#1\relax
1974     #3=#2\relax
1975   \else
1976     #3=#1\relax

```

1977 \fi}
 1978

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

```

\checkpageL 1979 \newif\ifl@dsamepage
\checkpageR 1980 \l@dsamepagetrue
1981 \newif\ifl@dpagefull
1982 \newcommand*\checkpageL{%
1983 \l@dpagefulltrue
1984 \l@dsamepagetrue
1985 \check@goal
1986 \ifdim\pagetotal<\ledthegoal
1987 \ifnum\numpagelinesL<\l@dminpagelines
1988 \else
1989 \l@dsamepagefalse
1990 \l@dpagefullfalse
1991 \fi
1992 \else
1993 \l@dsamepagefalse
1994 \l@dpagefulltrue
1995 \fi}
1996 \newcommand*\checkpageR{%
1997 \l@dpagefulltrue
1998 \l@dsamepagetrue
1999 \check@goal
2000 \ifdim\pagetotal<\ledthegoal
2001 \ifnum\numpagelinesR<\l@dminpagelines
2002 \else
2003 \l@dsamepagefalse
2004 \l@dpagefullfalse
2005 \fi
2006 \else
2007 \l@dsamepagefalse
2008 \l@dpagefulltrue
2009 \fi}
2010
```

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

```

2011 \newdimen\ledthegoal
2012 \ifshiftedpstarts
2013 \newcommand*\goalfraction{0.95}
2014 \else
```

```

2015      \newcommand*{\goalfraction}{0.9}
2016 \fi
2017
2018 \newcommand*{\check@goal}{%
2019   \ledthegoal=\goalfraction\pagegoal}
2020

\ifwrittenlinesL Booleans for whether line data has been written to the section file.
\ifwrittenlinesL 2021 \newif\ifwrittenlinesL
2022 \newif\ifwrittenlinesR
2023

\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.

2024 \newcommand*{\get@nextboxL}{%
2025   \ifvbox\namebox{1@dLcolrawbox\the\1@dpscL}% box is not empty

   The current box is not empty; do nothing.
2026   \else%                                box is empty

   The box is empty; check if enough lines (real and blank) have been output.
2027   \ifnum\usernamecount{1@dmaxlinesinpar\the\1@dpscL}>\@donetotallinesL
2028   \else

   Sufficient lines have been output.
2029   \ifwrittenlinesL
2030   \else

   Write out the number of lines done, and set the boolean so this is only done once.
2031   \@writelinesinparL
2032   \writtenlinesLtrue
2033   \fi
2034   \ifnum\1@dnumstartsL>\1@dpscL

   There are still unprocessed boxes. Recalculate the maximum number of lines
   needed, and move onto the next box (by incrementing \1@dpscL). If needed, restart
   the line numbering. Increment the pstartL counter.
2035   \writtenlinesLfalse
2036   \ifbypstart@
2037   \ifnum\value{pstartL}<\value{pstartLold}
2038   \else
2039     \global\line@num=0
2040     \resetprevline@
2041   \fi
2042   \fi
2043   \addtocounter{pstartL}{1}
2044   \global\pstartnumtrue
2045   \1@dcalc@maxoftwo{\the\usernamecount{1@dmaxlinesinpar\the\1@dpscL}}%
2046                     {\the\@donetotallinesL}%
2047                     {\usernamecount{1@dmaxlinesinpar\the\1@dpscL}}%
2048   \global\@donetotallinesL \z@

```

```

2049     \global\advance\l@dpscL \@ne
2050     \fi
2051     \fi
2052 \fi}

2053 \newcommand*{\get@nextboxR}{%
2054   \ifvbox\namebox{l@dRcolrawbox\the\l@dpscR}% box is not empty
2055   \else% box is empty
2056     \ifnum\usernamecount{l@dmaxlinesinpar\the\l@dpscR}>\@donetotallinesR
2057     \else
2058       \ifwrittenlinesR
2059       \else
2060         \@writelinesinparR
2061         \writtenlinesRtrue
2062       \fi
2063       \ifnum\l@dnumstartsR>\l@dpscR
2064       \writtenlinesRfalse
2065       \ifbypstart@R
2066         \ifnum\value{pstartR}<\value{pstartRold}
2067         \else
2068           \global\line@numR=0
2069           \resetprevline@
2070         \fi
2071       \fi
2072       \addtocounter{pstartR}{1}
2073       \global\pstartnumRtrue
2074       \l@dcalc@maxoftwo{\the\usernamecount{l@dmaxlinesinpar\the\l@dpscR}}%
2075         {\the\@donetotallinesR}%
2076       {\usernamecount{l@dmaxlinesinpar\the\l@dpscR}}}%
2077       \global\@donetotallinesR \z@
2078       \global\advance\l@dpscR \@ne
2079     \fi
2080   \fi
2081 \fi}
2082

```

25 The End

i/code_i

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>\&</code>	1529, 1530, 1534, 1551, 1572
<code>\@M</code>	1540
<code>\@adv</code>	<u>352</u> , 631, 632
<code>\@afterindentfalse</code>	754
<code>\@arabic</code>	209, 210, 803, 806
<code>\@astanza@line</code>	1550, 1556, <u>1559</u>
<code>\@auxout</code>	1328, 1340, 1646
<code>\@chapter</code>	755
<code>\@cs@linesinparL</code>	1822, <u>1933</u>
<code>\@cs@linesinparR</code>	1822, <u>1933</u>
<code>\@cs@linesonpageL</code>	1830, 1890, <u>1946</u>
<code>\@cs@linesonpageR</code>	1830, 1890, <u>1946</u>
<code>\@currentlabel</code>	838, 871
<code>\@donereallinesL</code>	
.	<u>915</u> , 944, 1784, 1786, 1836
<code>\@donereallinesR</code>	
.	<u>915</u> , 981, 1789, 1791, 1838
<code>\@donetotallinesL</code>	<u>915</u> ,
.	945, 948, 1837, 2027, 2046, 2048
<code>\@donetotallinesR</code>	<u>915</u> ,
.	982, 985, 1839, 2056, 2075, 2077
<code>\@insertR</code>	1282–1284, 1297–1299
<code>\@l</code>	<u>275</u> , 600
<code>\@l@dttempcnta</code>	425,
.	427, 429, 430, 434, 436, 438,
.	439, 1035, 1074, 1075, 1077,
.	1079, 1082, 1083, 1100–1104,
.	1106, 1113, 1118, 1122, 1130,
.	1135, 1139, 1172, 1175, 1177, 1181
<code>\@l@dttempcntb</code>	168, 170, 172, 1065,
.	1066, 1113, 1118, 1122, 1130,
.	1135, 1139, 1164, 1168, 1181,
.	1189–1191, 1193, 1213–1215,
.	1217, 1234–1236, 1238, 1369,
.	1371, 1373, 1444–1446, 1448,
.	1600–1604, 1608–1611, 1615–1618
<code>\@l@reg</code>	324
<code>\@l@regR</code>	275
<code>\@lab</code>	544, 1319, 1331, <u>1355</u>
<code>\@lock</code>	931, 1016
<code>\@lockR</code>	60, 297, 299, 301, 314,
.	459, 475, 476, 478, 479, 507,
.	508, 510, 968, 999, 1041, 1043,
.	1044, 1046, 1127, 1144, 1146, 1148
<code>\@lopL</code>	<u>568</u> , 1960
<code>\@lopR</code>	<u>568</u> , 1963
<code>\@nobreakfalse</code>	812, 845

- \@nobreaktrue 810, 814, 843, 847
 \@oldnobreak 810, 812, 843, 845, 885, 901
 \@pend 559, 1784
 \@pendR 559, 1789
 \@pstartsfalse 1757
 \@pstartstrue 1757
 \@ref 531, 604, 608
 \@ref@reg 557
 \@schapter 755
 \@set 384, 638, 639, 1731, 1735
 \@tag 668, 691, 1458
 \@temp 1637
 \@templ@d 1435, 1436
 \@writelinesinparL . . 1727, 1782, 2031
 \@writelinesinparR . . 1728, 1782, 2060
 \@writelinesonpageL . . 1860, 1862, 1959
 \@writelinesonpageR . . 1880, 1882, 1959
 \@xloop 1295
- A**
- \absline@num 418, 432, 451, 1007
 \absline@numR 58, 226, 277, 280, 283,
 415, 423, 444, 463, 497, 525,
 536, 990, 1027, 1028, 1065, 1281
 \actionlines@list
 267, 270, 418, 432, 451
 \actionlines@listR
 230, 245, 259, 262, 415,
 423, 444, 463, 497, 525, 1087, 1090
 \actions@list 271, 419, 439, 453, 455
 \actions@listR
 230, 246, 263, 416, 430, 446,
 448, 465, 474, 499, 506, 526, 1091
 \add@inserts 937
 \add@inserts@nextR 1270
 \add@insertsR 974, 1270
 \add@penaltiesL 943, 1291
 \add@penaltiesR 980, 1291
 \addtocontents 1647–1649
 \addtocounter
 887, 903, 1738, 1739, 2043, 2072
 \advancelabel@refs 1326, 1338
 \advanceline 630, 661
 \affixline@num 935
 \affixline@numR 972, 1097
 \affixpstart@numL 934, 1203
 \affixpstart@numR 971, 1203
 \affixside@note 938
 \affixside@noteR 975, 1422
 \appto 1428, 1429
- \apptocmd 1677
 \araw@textfalse 1769
 \araw@texttrue 1769
 astanza (environment) 9, 1532
 \AtBeginDocument 1351, 1622, 1657
- B**
- \ballast@count 1025, 1030
 \bbl@main@language 1673, 1674
 \bbl@set@language 1664, 1665
 \beginnumbering 7, 36, 761, 779, 817
 \beginnumberingR 49, 117, 779, 850
 \bfseries 803, 806
 \bypage@Rfalse 137, 152, 157
 \bypage@Rtrue 137, 147
 \bypstart@Rfalse 137, 148, 158
 \bypstart@Rtrue 137, 153
- C**
- \c@ballast 1030
 \c@chapter 100
 \c@chapterR 100
 \c@firstlinenumR 177, 1170
 \c@firstsublinenumR 181, 1165
 \c@linenumincrementR 177, 1170
 \c@page 600, 1911, 1917, 1920, 1927
 \c@pstartL 803
 \c@pstartR 806
 \c@section 101
 \c@sectionR 101
 \c@sublinenumincrementR 181, 1165
 \c@subsection 102
 \c@subsectionR 102
 \c@subsubsection 103
 \c@subsubsectionR 103
 \ch@ck@l@ckR 1097
 \ch@cksub@l@ckR 1097
 \ch@cksub@lockR 1166
 \chapter 741, 742, 750
 \chapterinpages 734, 742, 752
 \chardef 1529
 \check@goal 1985, 1999, 2011
 \check@pstarts
 1702, 1729, 1757, 1816, 1824, 1832
 \checkpageL 1843, 1857, 1979
 \checkpageR 1866, 1877, 1979
 \checkraw@text
 1708, 1725, 1769, 1840, 1886
 \cleardoublepage 1923
 \clearl@dleftpage 1865, 1909

`\clearl@drighthpage` 1885, [1909](#)
`\cleartoevenpage` [1909](#)
`\cleartol@devenpage` 1805, [1909](#)
`\closeout` 581, 587, 590, 594
`\columnrulewidth` 5, [1752](#)
`\Columns` 5, [1688](#)
`\columnseparator` 5, [1722](#), [1752](#)
`\content` 1381, 1395, 1409
`\correcthangingL` 940, [1510](#)
`\correcthangingR` 977, [1510](#)
`\countLline` [910](#), 921
`\countRline` [910](#), 958
`\critext` [666](#)
`\csexpandonce` 1384, 1388, 1398, 1402,
 1412, 1416, 1460, 1464, 1480, 1491
`\csuse` 1479, 1490

D

`\DeclareOption` 8, 9
`\def@tempb` 155
`\dimen` 617, 618, 622–624, 628
`\divide` 1102
`\do@actions` 1008
`\do@actions@fixedcodeR` [1034](#)
`\do@actions@nextR` [1034](#)
`\do@actionsR` 991, [1034](#)
`\do@ballast` 1009
`\do@ballastR` 992, [1025](#)
`\do@insidelineLhook` 940, [952](#)
`\do@insidelineRhook` [952](#), 977
`\do@lineL` [920](#), 1712, 1716, 1848
`\do@lineLhook` 925, [952](#)
`\do@lineR` [957](#), 1713, 1718, 1869
`\do@lineRhook` [952](#), 962
`\do@lockoff` [494](#)
`\do@lockoffL` 518
`\do@lockoffR` [494](#)
`\do@lockon` [459](#)
`\do@lockonL` 491
`\do@lockonR` [459](#)
`\dolistloop` 1433
`\dummy@ref` 540

E

`\edfont@info` 717, 720, 726, 729
`\edlabel` [1317](#)
`\edtext` [689](#)
`\eledmac@error` 21, 24, 27, 29
`\eledmac@warning` 1434

`\empty` .. 81, 84, 259, 267, 679, 702,
 715, 724, 826, 859, 1087, 1169,
 1177, 1272–1274, 1285, 1296,
 1320, 1332, 1934, 1940, 1947, 1953
`\end@lemmas` 679, 680, 702, 703
`\endashchar` 1307
`\endgraf` 881, 897, 1696, 1808
`\endline@num` 547, 553
`\endlock` [650](#), 1538, 1547, 1552
`\endnumbering` 7, 39, [74](#), 121, 780
`\endnumberingR` 52, [74](#), 106, 116, 129, 780
`\endpage@num` 546, 553
`\endstanzaextra` 1554
`\endsub` [617](#)
`\endsubline@num` 548, 554
environments:

 astanza 9, [1532](#)
 Leftside 6, [759](#)
 pages 5, [734](#)
 pairs 5, [734](#)
 Rightside 6, [777](#)
`\extensionchars` . 47, 66, 112, 126, 134

F

`\f@x@l@cksR` [1097](#)
`\first@linenum@out@Rfalse` .. 576, 582
`\first@linenum@out@Rtrue` [576](#)
`\firstlinenum` 6, [186](#)
`\firstsublinenum` 6, [186](#)
`\fix@page` 320, [327](#)
`\flag@end`
 . [601](#), 675, 685, 686, 698, 708, 709
`\flag@start`
 . [601](#), 674, 675, 686, 697, 698, 709
`\flush@notes` 1741, 1895
`\flush@notesR` [1294](#), 1742, 1896
`\footnotelang@lua` 1474, 1485
`\footnotelang@poly` 1478, 1489
`\fullstop` . 222, 1304, 1306, 1308, 1310

G

`\get@linelistfile` 255
`\get@nextboxL` 1856, [2024](#)
`\get@nextboxR` 1876, [2024](#)
`\getline@numL` 930, 1006
`\getline@numR` 967, [989](#)
`\getlinesfrompagelistL`
 1828, 1888, [1946](#)
`\getlinesfrompagelistR`
 1829, 1889, [1946](#)

\getlinesfromparlistL ... 1820, 1933
 \getlinesfromparlistR ... 1821, 1933
 \gl@p 262, 263,
 270, 271, 680, 703, 719, 728,
 1090, 1091, 1278, 1282, 1297,
 1323, 1335, 1937, 1943, 1950, 1956
 \goalfraction 6, 2011

H

\hangingsymbol 10, 1501, 1507
 \hb@xt@ ... 933, 940, 947, 970, 977,
 984, 1720, 1851, 1853, 1872, 1874
 \hsize 836,
 869, 1720, 1851, 1853, 1872, 1874

I

\if@filesw 1645
 \if@firstcolumn 1183, 1207, 1228, 1438
 \if@ledgroup 586
 \if@nobreak 809, 842
 \if@pstarts 1703, 1757, 1817, 1833
 \if@RTL 675, 686, 698, 709
 \ifaraw@text ... 1710, 1769, 1842, 1887
 \ifautopar 835, 868
 \ifbypage@ 343
 \ifbypage@R 137, 333, 1069
 \ifbypstart@ 561, 1730, 2036
 \ifbypstart@R ... 137, 565, 1734, 2065
 \ifdim 618, 622, 624,
 628, 1851, 1872, 1913, 1986, 2000
 \iffirst@linenum@out@R 576, 580
 \ifinserthangingsymbol .. 1499, 1513
 \ifinserthangingsymbolR
 1497, 1505, 1523
 \ifinstanzaL 757, 757, 1500, 1512
 \ifinstanzaR 757, 758, 1506, 1522
 \ifl@d@dash 1307
 \ifl@d@elin 1309, 1310
 \ifl@d@esl 1310
 \ifl@d@pnum 1304, 1308
 \ifl@d@ssub 1306
 \ifl@dpagefull 1859, 1879, 1979
 \ifl@dpaging 11, 1511, 1521
 \ifl@dpairing 11, 78
 \ifl@dsamelang 1633, 1711
 \ifl@dsamepage 1846, 1868, 1979
 \ifl@dskipnumber 1160
 \ifl@dusedbabel 1631
 \iflabelpstart 838, 871
 \ifledplinenum 1305

\ifledRcol 11, 169, 191,
 195, 199, 203, 242, 257, 321,
 330, 354, 368, 385, 402, 414,
 422, 443, 488, 515, 524, 533,
 602, 612, 619, 625, 631, 638,
 646, 651, 655, 660, 670, 693,
 714, 1318, 1356, 1370, 1383,
 1397, 1411, 1459, 1472, 1668, 1678
 \ifluatex 1473, 1484
 \ifnoteschanged@ 88
 \ifnumberedpar@ ... 819, 852, 877,
 893, 1382, 1396, 1410, 1457, 1471
 \ifnumbering 37, 142, 815, 874
 \ifnumberingR ... 50, 75, 108, 848, 890
 \ifnumberline 993, 1010, 1159
 \ifnumberpstart ... 835, 868, 886, 902
 \ifnumequal 1426
 \ifnumgreater 1434
 \ifodd 1193, 1217,
 1238, 1448, 1911, 1917, 1920, 1927
 \ifpst@rtedL 32, 823
 \ifpst@rtedR 32, 856
 \ifpstartnum 1248, 1253
 \ifpstartnumR 1203
 \ifshiftedpstarts 5, 1850, 1871, 2012
 \ifsidepstartnum 835, 868, 1205, 1226
 \ifsublines@ 220,
 309, 353, 386, 393, 424, 433,
 445, 452, 464, 498, 552, 554,
 994, 1011, 1076, 1163, 1358, 1362
 \ifvbox 922, 959, 1773, 1776, 2025, 2054
 \ifvmode 1325, 1337
 \ifwrittenlinesL 2021, 2029
 \ifwrittenlinesR 2022, 2058
 \initnumbering@reg 45
 \initnumbering@sectcmd 70
 \initnumbering@sectcountR 71, 95
 \insert@count 530, 608, 671,
 694, 1390, 1404, 1418, 1466, 1493
 \insert@countR 531, 604, 670,
 693, 1386, 1400, 1414, 1462, 1482
 \inserthangingsymbolfalse 931
 \inserthangingsymbolL 940, 1497
 \inserthangingsymbolR 977, 1497
 \inserthangingsymbolRfalse 968
 \inserthangingsymbolRtrue 968
 \inserthangingsymboltrue 931
 \insertlines@listR
 81, 230, 244, 536, 1274, 1278

- \inserts@list 825, 1389, 1403, 1417, 1465, 1492
 - \inserts@listR 858, 1269, 1272, 1282, 1296, 1297, 1385, 1399, 1413, 1461, 1481
 - \istanzaLfalse 1749, 1902
 - \istanzaLtrue 768
 - \istanzaRfalse 1750, 1903
 - \istanzaRtrue 790
 - \interlinepenalty 1540
 - \itemcount@ 1424, 1426, 1431, 1434
- L**
- \l@d@nums 717, 720, 726, 729
 - \l@d@set 401, 646, 647
 - \l@dbbl@set@language 1642, 1665
 - \l@dbfnote 1456
 - \l@dc@maxchunks 831, 833, 864, 866, 1590, 1600, 1608, 1615
 - \l@dcalc@maxoftwo 1822, 1966, 2045, 2074
 - \l@dcalc@minoftwo .. 1830, 1890, 1966
 - \l@dcalcnun 1097
 - \l@dchecklang 1635, 1709
 - \l@dchset@num 276, 279, 401
 - \l@dcsnote 1379
 - \l@dcsnotetext 1433, 1436
 - \l@emptyd@ta 926, 963
 - \l@dend@stuff ... 48, 67, 113, 127, 135
 - \l@dgetline@margin 167
 - \l@dgetsidenote@margin 1368
 - \l@dld@ta 936, 973, 1184, 1196, 1208, 1220, 1229, 1241
 - \l@dleftbox 907, 932, 947, 1721, 1851, 1853
 - \l@dlinenumR 212
 - \l@dlsn@te 939, 976
 - \l@dlsnote 1379
 - \l@dmake@labels 1341
 - \l@dmake@labelsR 1329, 1345
 - \l@dminpagelines 1793, 1831, 1891, 1987, 2001
 - \l@dnumpstartsL . 41, 830, 831, 833, 835, 1594, 1626, 1691, 1692, 1746, 1760, 1802, 1803, 1900, 2034
 - \l@dnumpstartsR . 54, 863, 864, 866, 868, 1594, 1627, 1691, 1692, 1747, 1763, 1802, 1803, 1901, 2063
 - \l@doldbbl@set@language 1664
 - \l@doldselectlanguage 1663, 1667, 1672
 - \l@dpagefullfalse 1979
 - \l@dpagefulltrue 1979
 - \l@dpageingfalse 13, 736, 749
 - \l@dpageingtrue 744
 - \l@dpairingfalse 11, 738, 748
 - \l@dpairingtrue 735, 743
 - \l@dpscL 922, 927, 1596, 1628, 1700, 1706, 1744, 1760, 1773, 1812, 1818, 1823, 1826, 1834, 1898, 2025, 2027, 2034, 2045, 2047, 2049
 - \l@dpscR 959, 964, 1597, 1629, 1701, 1707, 1745, 1763, 1776, 1813, 1819, 1827, 1835, 1899, 2054, 2056, 2063, 2074, 2076, 2078
 - \l@drd@ta 940, 977, 1186, 1194, 1210, 1218, 1231, 1239
 - \l@drightbox 907, 969, 984, 1723, 1872, 1874
 - \l@drsn@te 941, 978
 - \l@drsnote 1379
 - \l@dsamelangfalse 1633, 1636
 - \l@dsamelangtrue 1633, 1639
 - \l@dsamepagefalse 1979
 - \l@dsamepagetrue 1979
 - \l@dsetupmaxlinecounts .. 1607, 1624
 - \l@dsetupprawboxes 1599, 1623
 - \l@dskipnumberfalse 1161
 - \l@dskipnumbertrue 1057
 - \l@dunhbox@line 940, 977
 - \l@dusedbabelfalse 1631, 1660
 - \l@dusedbabeltrue 1631, 1662
 - \l@duselanguage 1652, 1715, 1717, 1844, 1867
 - \l@dzeromaxlinecounts ... 1607, 1625
 - \l@dzeropenalties 880, 896, 1695, 1807
 - \l@pscL 1596
 - \l@pscR 1596
 - \label@refs 1321, 1323, 1329, 1333, 1335, 1341
 - \labelref@list 1332, 1335, 1363
 - \labelref@listR 1315, 1320, 1323, 1359
 - \languagename .. 1643, 1644, 1646–1649
 - \last@page@num 341, 347
 - \last@page@numR 327
 - \lastbox 929, 966
 - \lastskip 617, 623
 - \Lcolwidth 5, 6, 15, 745, 836, 933, 947
 - \led@err@BadLeftRightPstarts 23, 1692, 1803
 - \led@err@LeftOnRightPage ... 26, 1921

- \led@err@LineationInNumbered ... 143
 - \led@err@NumberingNotStarted ... 92
 - \led@err@NumberingShouldHaveStarted
..... 115
 - \led@err@NumberingStarted 38, 51
 - \led@err@PendNoPstart 878, 894
 - \led@err@PendNotNumbered ... 875, 891
 - \led@err@PstartInPstart ... 820, 853
 - \led@err@PstartNotNumbered . 816, 849
 - \led@err@RightOnLeftPage ... 26, 1928
 - \led@err@TooManyPstarts 20, 832, 865
 - \led@mess@NotesChanged 89
 - \led@mess@SectionContinued
..... 111, 125, 133
 - \led@warn@BadAction 1059
 - \led@warn@BadAdvancelineLine 371, 377
 - \led@warn@BadAdvancelineSubline .
..... 357, 363
 - \led@warn@BadLineation 160
 - \led@warn@BadSetline 636
 - \led@warn@BadSetlinenum 644
 - \led@warn@DuplicateLabel 1347
 - \ledllfill 940, 977
 - \ledRcolfalse 14, 760, 792
 - \ledRcoltrue 778
 - \ledrlfill 940, 977
 - \ledsavedprintlines 8, 1302
 - \ledstrutL 1851, 1853, 1906
 - \ledstrutR 1872, 1874, 1906
 - \ledthegoal 1986, 2000, 2011
 - \leftlinenumR 212, 1184, 1196
 - \leftpstartnumL 1203
 - \leftpstartnumR 1203
 - Leftside (environment) 6, 759
 - \Leftsidehook 766, 772
 - \Leftsidehookend 771, 772
 - \line@list 724, 728
 - \line@list@stuff 47, 126
 - \line@list@stuffR .. 66, 112, 134, 578
 - \line@listR . 84, 230, 243, 554, 715, 719
 - \line@margin 172, 1213
 - \line@marginR 165, 1189, 1234
 - \line@num . 344, 375, 376, 378, 396,
407, 408, 436, 561, 1017, 1361, 2039
 - \line@numR 59, 219, 226,
281, 315, 334, 369, 370, 372,
389, 403, 404, 427, 547, 551,
565, 1000, 1070, 1079, 1168,
1170, 1172, 1173, 1357, 1434, 2068
 - \lineation 787
 - \lineationR 141, 787
 - \linenum@out 607,
615, 620, 626, 632, 639, 647,
652, 656, 1331, 1731, 1784, 1960
 - \linenum@outR
. 575, 581, 583, 587, 588, 590,
591, 594, 595, 600, 603, 613,
619, 625, 631, 638, 646, 651,
655, 660, 1319, 1735, 1789, 1963
 - \linenumberlist 1169, 1173
 - \linenumincrement 6, 186
 - \linenummargin 165
 - \linenumr@p 1305, 1309, 1357, 1361
 - \linenumrepR 209, 219
 - \linenumsep
. 214, 216, 1250, 1253, 1262, 1265
 - \linesinpar@listL
..... 235, 251, 562, 1934, 1937
 - \linesinpar@listR
..... 235, 247, 566, 1940, 1943
 - \linesonpage@listL 252, 570, 1947, 1950
 - \linesonpage@listR 248, 573, 1953, 1956
 - \list@clear
. 243–248, 251, 252, 254, 825, 858
 - \list@clearing@reg 250
 - \list@create
... 230–233, 235–237, 1269, 1315
 - \lock@disp 1129, 1133, 1138
 - \lock@off 485, 486, 494, 655, 656
 - \lock@on 651, 652
- M
- \managestanza@modulo 1566
 - \maxchunks 4, 1590
 - \maxlinesinpar@list 235, 254
 - \memorydump 8, 765, 783
 - \memorydumpL 120, 765
 - \memorydumpR 120, 783
 - \message 46, 65
 - \multiply 1103
- N
- \n@num 522, 660
 - \n@num@reg 528
 - \namebox 922, 927, 959,
964, 1574, 1773, 1776, 2025, 2054
 - \NeedsTeXFormat 2
 - \new@line 940
 - \new@lineR 599, 977
 - \newbox 798, 907, 908, 1575

- \newcounter 95–98, 177,
179, 181, 183, 801, 802, 804, 805
 - \newif . 5, 12, 33, 137, 138, 576, 757,
758, 1257, 1497, 1631, 1633,
1757, 1769, 1979, 1981, 2021, 2022
 - \newnamebox 1574, 1602, 1603
 - \newnamecount 1585, 1610
 - \newwrite 575
 - \next@action 271
 - \next@actionline 268, 270
 - \next@actionlineR
. 260, 262, 1028, 1066, 1088, 1090
 - \next@actionR 263, 1029,
1067, 1068, 1073, 1074, 1082, 1091
 - \next@insert 826
 - \next@insertR
859, 1273, 1276, 1278, 1281, 1285
 - \next@page@num 348, 419
 - \next@page@numR 63, 284, 286, 338, 416
 - \no@expands 668, 691
 - \normal@pars 77, 829, 862
 - \normalbfnoteX 1470
 - \noteschanged@true
..... 82, 85, 716, 725, 1275
 - \num@lines 881, 1696, 1808
 - \num@linesR 797, 897, 1697, 1809
 - \numberedpar@true 837, 870
 - \numberingRfalse 76
 - \numberingRtrue 56, 106, 130
 - \numberingtrue 43, 122
 - \numberpstartfalse 9
 - \numberpstarttrue 9
 - \numdef 1424, 1431
 - \numlabfont 219
 - \numpagelinesL
.... 1793, 1849, 1860, 1864, 1987
 - \numpagelinesR
.... 1793, 1870, 1880, 1884, 2001
- O**
- \oldchapter 741, 750
 - \oldstanza 767, 768, 770, 789, 790, 793
 - \one@line 927, 929, 940
 - \one@lineR 797, 964, 966, 977
 - \openout 583, 588, 591, 595
- P**
- \p@pstartL 839
 - \p@pstartR 872
 - \page@action 285, 413, 541
 - \page@num 266, 346, 1215, 1446
 - \page@numR 239, 258, 336,
546, 551, 1068, 1191, 1236, 1434
 - \pagegoal 2019
 - \Pages 5, 1797
 - pages (environment) 5, 734
 - \pagetotal 1913, 1986, 2000
 - pairs (environment) 5, 734
 - \par@line 882, 1698, 1810
 - \par@lineR 797, 898, 1699, 1811
 - \pausenumbering 781
 - \pausenumberingR 105, 781
 - \pend 6, 764, 786, 821, 1553
 - \pendL 764, 874
 - \pendR 786, 854, 890
 - \prevgraf
. 881, 897, 1696, 1697, 1808, 1809
 - \printlines 1313
 - \printlinesR 8, 1302
 - \ProcessOptions 10
 - \protected@csxdef 1479, 1490
 - \protected@edef 838, 871
 - \protected@write ... 1328, 1340, 1646
 - \ProvidesPackage 3
 - \pst@rtedLfalse 32, 42
 - \pst@rtedLtrue 123, 827
 - \pst@rtedRfalse 34, 55, 79
 - \pst@rtedRtrue 109, 131, 860
 - \pstart 6, 21, 25, 762, 785, 1555
 - \pstartL 762, 800
 - \pstartnumfalse 1250, 1255
 - \pstartnumRfalse 1262, 1267
 - \pstartnumRtrue 1258, 1705, 2073
 - \pstartnumtrue 1704, 2044
 - \pstartR 785, 800
- R**
- \Rcolwidth 5, 6, 15, 746, 869, 970, 984
 - \read@linelist 241, 579
 - \rem@inder 1173, 1175–1177
 - \resetprevline@ 1732, 1736, 2040, 2069
 - \resumenumbering 782
 - \resumenumberingR 105, 782
 - \rightlinenumR 212, 1186, 1194
 - \rightpstartnumL 1203
 - \rightpstartnumR 1203
 - Rightside (environment) 6, 777
 - \Rightsidehook 772, 788
 - \Rightsidehookend 772, 794
 - \rlap 1186, 1194, 1210, 1218, 1231, 1239

\Rlineflag 8, 207, 219, 1305, 1309, 1349
 \rule 1753

S

\sc@n@list 1174, 1176
 \secdef 755
 \section@num 44, 46, 47, 124–126
 \section@numR
 . 30, 57, 65, 66, 110–112, 132–134
 \select@language ... 1644, 1646–1649
 \selectlanguage 1652
 \set@line 669, 692, 713
 \set@line@action
 278, 382, 391, 398, 421, 543
 \setl@dlp@rbox 1439, 1451
 \setl@drp@rbox 1441, 1449
 \setline 634
 \setlinenum 642
 \setnamebox 835, 868, 1574
 \setprintlines 1303
 \shiftedpstartfalse 7
 \shiftedpstarttrue 6, 8, 9
 \shiftedversesfalse 7
 \shiftedversestrue 6
 \showlemma 678, 701
 \sidenote@margin 1373, 1377
 \sidenote@marginR 1366, 1444
 \sidenotecontent@
 1423, 1428, 1429, 1439, 1441, 1451
 \sidenotecontent@t 1449
 \sidenotemargin 1366
 \sidenotesep 1429
 \skip@lockoff 486, 494
 \skipnumbering 9, 659
 \skipnumbering@reg 663
 \smash 1753
 \splittopskip 924, 961
 \stanza ... 767, 768, 770, 789, 790, 793
 \stanza@count 1535, 1549, 1561
 \stanza@hang 1537, 1569
 \stanza@modulo 1535, 1564
 \stanzaindentbase
 1515, 1525, 1562, 1565
 \startlock 650
 \startstanzahook 1533
 \startsub 617
 \sub@action 294, 442, 542
 \sub@change 64, 288, 289, 295
 \sub@lock 1012

\sub@lockR 61, 303, 305, 307,
 310, 460, 466, 467, 469, 470,
 500, 501, 503, 995, 1049, 1051,
 1052, 1054, 1110, 1150, 1152, 1154
 \sub@off 625, 626
 \sub@on 619, 620
 \subline@num 221, 344, 361,
 362, 364, 394, 434, 1013, 1018, 1362
 \subline@numR 222,
 226, 311, 315, 334, 355, 356,
 358, 387, 425, 548, 552, 996,
 1001, 1070, 1077, 1164, 1165, 1358
 \sublinenumincrement 6, 186
 \sublinenumr@p . 1306, 1310, 1358, 1362
 \sublinenumrepR 209, 222
 \sublines@false 62, 292, 1039
 \sublines@true 290, 1037
 \sublock@disp 1112, 1116, 1121
 \symplinenum 1305
 \sza@penalty 1544, 1548

T

\textwidth 16, 18, 745, 746
 \theledlanguageL 1637, 1652, 1715, 1844
 \theledlanguageR 1637, 1652, 1717, 1867
 \thepage 600, 1329, 1341
 \thepstart 763, 784
 \thepstartL
 . 9, 763, 803, 835, 839, 1249, 1254
 \thepstartR
 . 9, 784, 806, 868, 872, 1261, 1266
 \thr@@ .. 469, 478, 501, 508, 1044, 1052
 \topskip 1913

U

\unhbox 1579,
 1721, 1723, 1851, 1853, 1872, 1874
 \unhnamebox 1574
 \unvbox 929, 966, 1581
 \unvnamebox 1574
 \usenamecount
 . 1536, 1543, 1585, 1617, 1823,
 2027, 2045, 2047, 2056, 2074, 2076

V

\value 824, 857, 1560,
 1689, 1690, 1798, 1799, 2037, 2066
 \vbadness 923, 960
 \vbfnoteX 1480, 1491
 \vbox 835, 868

<code>\vl@dbfnote</code>	1460, 1464	<code>\xpg@main@language</code>	1682, 1683
<code>\vl@dcsnote</code>	1412, 1416	<code>\xpg@set@language</code>	1677, 1681
<code>\vl@dlsnote</code>	1384, 1388	<code>\xright@appenditem</code>	
<code>\vl@drsnote</code>	1398, 1402 415, 416, 418, 419, 423,	
<code>\vsplit</code>	927, 964	430, 432, 439, 444, 446, 448,	
W			
<code>\wd</code>	940, 977	451, 453, 455, 463, 465, 474,	
<code>\writtenlinesLfalse</code>	1814, 2035	497, 499, 506, 525, 526, 536,	
<code>\writtenlinesLtrue</code>	2032	550, 562, 566, 570, 573, 1357,	
<code>\writtenlinesRfalse</code>	1815, 2064	1361, 1384, 1388, 1398, 1402,	
<code>\writtenlinesRtrue</code>	2061	1412, 1416, 1460, 1464, 1480, 1491	
X			
<code>\x@lemma</code>	680–682, 703–705	Z	
		<code>\zz@@@</code>	1321, 1333

Change History

v0.1		pstart.	61
General: First public release	1	<code>\get@nextboxR</code> : Change <code>\get@nextboxL</code>	
v0.10		and <code>\get@nextboxR</code> to allow to	
General: <code>\edlabel</code> commands on		disable line numbering (like in	
the right side are now correctly		eledmac 0.15).	70
indicated.	1	Pstart number can be printed in	
<code>\edlabel</code> commands which start		side	70
a paragraph are now put in the			
right place.	1	v0.12	
v0.11		General: New new management of	
General: Change <code>\do@lineL</code> and		hangingsymbol insertion, pre-	
<code>\do@lineR</code> to allow line num-		venting undesirable insertions.	54
bering by pstart (like in eledmac		v0.2	
0.15).	39	General: Added section of babel re-	
Lineation can be by pstart (like		lated code	58
in eledmac 0.15).	16	Fix babel problems	1
New management of hang-		<code>\Columns</code> : Added <code>\l@dcchecklang</code>	
ingsymbol insertion, preventing		and <code>\l@duselanguage</code> to	
undesirable insertions.	54	<code>\Columns</code>	61
Prevent shift of column separator		<code>\Pages</code> : Added <code>\l@duselanguage</code>	
when a verse is hanged	54	to <code>\Pages</code>	65
<code>\affixline@numR</code> : Changed		v0.3	
<code>\affixline@numR</code> to allow to		General: Added <code>\do@lineLhook</code>	
disable line numbering (like in		and <code>\do@lineRhook</code>	40
eledmac 0.15).	43	Reorganize for ledarab	1
<code>\Columns</code> : Line numbering by		<code>\affixline@numR</code> : Changed	
		<code>\affixline@numR</code> to match new	

eledmac	43	v0.3a	
\do@actions@nextR:	Used	\line@marginR:	Don't just
\do@actions@fixedcode	in	set	\line@marginR in
\do@actionsR	42	\linenummargin	17
\do@lineL: Added \do@lineLhook		v0.3b	
to \do@lineL	39	\Pages: Added \l@dminpagelines	
Simplified \do@lineL by using		calculation for succeeding page	
macros for some common code	39	pairs	66
\do@lineR: Changed \do@lineR		v0.4	
similarly to \do@lineL	40	General: No more ledparpatch. All	
Leftside: Added hooks into Left-		patches are now in the main	
side environment	34	file.	1
\flag@end: Removed extraneous		v0.5	
spaces from \flag@end	30	General: Corrections about	
\ifledRcol: Moved \ifl@dpairing		\section and other titles in	
to eledmac	13	numbered sections	1
\ifpst@rtedR: Moved \ifpst@rtedL		v0.6	
to eledmac	14	General: Be able to us \chapter in	
\l@dlinenumR: Simplified		parallel pages.	1
\leftlinenumR and \rightlinenumR		v0.7	
by introducing \l@dlinenumR	19	General: Option 'shiftedverses'	
\l@dnumpstartsR: Moved		which make there is no blank	
\l@dnumpstartsL to eledmac	57	between two parallel verses with	
\ledsavedprintlines: Simplified		inequal length.	1
\printlinesR by using		v0.8	
\setprintlines	49	General: Possibility to have a sym-	
\ledstrutR: Added \ledstrutL and		bol on each hanging of verses,	
\ledstrutR	67	like in the french typogra-	
\normalbfnoteX: Removed		phy. Redefine the commande	
extraneous spaces from		\hangingsymbol to define the	
\normalbfnoteX	53	character.	1
\Pages: Added \ledstrutL to		v0.9	
\Pages	65	General: Possibility to number	
Added \ledstrutR to \Pages	66	\pstart.	9
\Rightsidehookend: Added		Possibility to number the	
\Leftsidehook, \Leftsidehookend,		pstart with the commands	
\Rightsidehook and \Rightsidehookend		\numberpstarttrue.	1
	35	\ifledRcol: Moved \iflledRcol	
\sublinenumrepR: Added		and \ifnumberingR to eledmac	13
\linenumrepR and \sublinenumrepR		v0.9.1	
	19	General: The numbering of the	
v0.3.a		pstarts restarts on each	
General: Minor \linenummargin		\beginnumbering.	1
fix	1	v0.9.2	
v0.3.b		General: Debug : with \Columns,	
General: Improved parallel page		the hanging indentation now	
balancing	1	runs on the left columns and the	
v0.3.c		hanging symbol is shown only	
General: Compatibilty with Poly-		when \stanza is used.	1
glossia	1		

v0.9.3	General: <code>\thepstartL</code> and <code>\thepstartR</code> use now <code>\bfseries</code> and not <code>\bf</code> , which is deprecated and makes conflicts with memoir class.	1	v1.3.2	General: Debug with some classes.	1
v1.0	General: Compatibility with eledmac. Change name to eledpar.	1	v1.3.3	<code>\l@dbfnote</code> : Spurious space with footnote in right column.	53
	Debug in lineation by pstart	16		<code>\l@dcsnote</code> : Debug on the left notes of the right column.	51
v1.0.1	General: Correction on <code>\numberonlyfirstline</code> with lineation by pstart or by page.	1	v1.3.4	<code>\l@dcsnote</code> : Allow to use commands in sidenotes, like it was introduced by eledmac 1.0.	51
v1.1	General: <code>Shiftedverses</code> becomes <code>shiftedpstarts</code>	1	v1.3.5	<code>\normalbfnoteX</code> : Allows one to redefine <code>\thefootnoteX</code> with alph when some packages are loaded.	53
v1.1.2	<code>\affixside@noteR</code> : Remove spurious space between line number and line content	52	v1.4	General: Added <code>\do@insidelineLhook</code> and <code>\do@insidelineRhook</code>	40
v1.2	General: Support for <code>\led<section></code> commands in parallel texts.	1	v1.4.1	<code>\normalbfnoteX</code> : Fix bug with normal familiar footnotes when mixing RTL and LTR text.	53
v1.2.1	<code>\initnumbering@sectcountR</code> : For the right section, the counter is defined only once.	15		<code>astanza</code> : Enable the use of <code>stanzaindent</code> s repetition within <code>astanza</code> environment.	55
v1.3	General: Manage RTL language.	32	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