

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 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 v1.2, last revised 2012/10/08.

[†]herries dot press at earthlink dot net

[‡]maieul at maieul dot net

7 Verse	9
8 Implementation overview	12
9 Preliminaries	12
9.1 Messages	13
10 Sectioning commands	13
11 Line counting	16
11.1 Choosing the system of lineation	16
11.2 Line-number counters and lists	19
11.3 Reading the line-list file	20
11.4 Commands within the line-list file	21
11.5 Writing to the line-list file	29
12 Marking text for notes	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	38
14.3 Line and page number computation	40
14.4 Line number printing	43
14.5 Pstart number printing in side	45
14.6 Add insertions to the vertical list	46
14.7 Penalties	47
14.8 Printing leftover notes	48
15 Footnotes	48
15.1 Normal footnote formatting	48
16 Cross referencing	49
17 Side notes	50
18 Familiar footnotes	52
19 Verse	53
20 Naming macros	55
21 Counts and boxes for parallel texts	55
22 Fixing babel	57
23 Parallel columns	59

<i>List of Figures</i>	3
24 Parallel pages	62
25 The End	70
References	71
Index	71
Change History	79

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 **\Rcolwidth** 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 **\columnseparator** 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 be 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}[2012/10/08 v1.2 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` For we want the numbering of the section commands in the right side to be not continous with the numbering of the rightside we switch the \LaTeX counter in `\numberingR`.

```

95 \newcommand{\initnumbering@sectcountR}{
96 \newcounter{chapterR}
97 \newcounter{sectionR}
98 \newcounter{subsectionR}
99 \newcounter{subsubsectionR}
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` 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

```

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

```

207 \newcommand*{\Rlineflag}{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}\Rlineflag%
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 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 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@outR` 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     \closeout\linenum@outR
586     \openout\linenum@outR=#1
587   \fi}
588

```

`\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.

```

589 \newcommand*{\new@lineR}{%
590   \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 send the `\@ref` command to the line-list file.

```

591 \renewcommand*{\flag@start}{%
592   \ifledRcol
593     \edef\next{\write\linenum@outR{%
594       \string\@ref[\the\insert@countR][ ]}%
595     \next

```

```

596 \else
597   \edef\next{\write\linenum@out{%
598             \string\@ref[\the\insert@count] []}}%
599   \next
600 \fi}
601 \renewcommand*{\flag@end}{%
602   \ifledRcol
603     \write\linenum@outR{[]}%
604   \else
605     \write\linenum@out{[]}%
606   \fi}

```

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

```

607 \renewcommand*{\startsub}{\dimen0\lastskip
608   \ifdim\dimen0>0pt \unskip \fi
609   \ifledRcol \write\linenum@outR{\string\sub@on}%
610   \else      \write\linenum@out{\string\sub@on}%
611   \fi
612   \ifdim\dimen0>0pt \hskip\dimen0 \fi}
613 \def\endsub{\dimen0\lastskip
614   \ifdim\dimen0>0pt \unskip \fi
615   \ifledRcol \write\linenum@outR{\string\sub@off}%
616   \else      \write\linenum@out{\string\sub@off}%
617   \fi
618   \ifdim\dimen0>0pt \hskip\dimen0 \fi}
619

```

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

```

620 \renewcommand*{\advanceline}[1]{%
621   \ifledRcol \write\linenum@outR{\string\@adv[#1]}%
622   \else      \write\linenum@out{\string\@adv[#1]}%
623   \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.

```

624 \renewcommand*{\setline}[1]{%
625   \ifnum#1<\z@
626     \led@warn@BadSetline
627   \else
628     \ifledRcol \write\linenum@outR{\string\@set[#1]}%
629     \else      \write\linenum@out{\string\@set[#1]}%
630     \fi
631   \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.

```

632 \renewcommand*{\setlinenum}[1]{%

```

```

633 \ifnum#1<\z@
634   \led@warn@BadSetlinenum
635 \else
636   \ifledRcol \write\linenum@outR{\string\l@d@set[#1]}
637   \else      \write\linenum@out{\string\l@d@set[#1]} \fi
638 \fi}
639

```

`\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.

```

640 \renewcommand*{\startlock}{%
641   \ifledRcol \write\linenum@outR{\string\lock@on}%
642   \else      \write\linenum@out{\string\lock@on}%
643 \fi}
644 \def\endlock{%
645   \ifledRcol \write\linenum@outR{\string\lock@off}%
646   \else      \write\linenum@out{\string\lock@off}%
647 \fi}
648

```

`\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.

```

649 \renewcommand*{\skipnumbering}{%
650   \ifledRcol \write\linenum@outR{\string\n@num}%
651   \advanceline{-1}%
652 \else
653   \skipnumbering@reg
654 \fi}
655

```

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.

```

656 \long\def\critext#1#2/{\leavevmode
657   \begingroup
658     \renewcommand{\@tag}{\no@expands #1}%
659     \set@line
660     \ifledRcol \global\insert@countR \z@
661     \else      \global\insert@count \z@ \fi
662     \ignorespaces #2\relax
663     \flag@start
664   \endgroup
665   \showlemma{#1}%
666   \ifx\end@lemmas\empty \else
667     \gl@p\end@lemmas\to\x@lemma
668     \x@lemma
669     \global\let\x@lemma=\relax
670   \fi
671   \flag@end}

```

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

```

672 \renewcommand{\edtext}[2]{\leavevmode
673   \begingroup
674     \renewcommand{\@tag}{\no@expands #1}%
675     \set@line
676     \ifledRcol \global\insert@countR \z@
677     \else      \global\insert@count \z@ \fi
678     \ignorespaces #2\relax
679     \flag@start
680   \endgroup
681   \showlemma{#1}%
682   \ifx\end@lemmas\empty \else
683     \gl@p\end@lemmas\to\x@lemma
684     \x@lemma
685     \global\let\x@lemma=\relax
686   \fi
687   \flag@end}
688

```

`\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`.

```

689 \renewcommand*{\set@line}{%
690   \ifledRcol
691     \ifx\line@listR\empty
692       \global\noteschanged@true
693       \xdef\l@d@nums{000|000|000|000|000|000|\edfont@info}%
694     \else
695       \gl@p\line@listR\to\@tempb
696       \xdef\l@d@nums{\@tempb|\edfont@info}%

```

```

697     \global\let\@tempb=\undefined
698     \fi
699   \else
700     \ifx\line@list\empty
701       \global\noteschanged@true
702       \xdef\l@d@nums{000|000|000|000|000|000|\edfont@info}%
703     \else
704       \gl@p\line@list\to\@tempb
705       \xdef\l@d@nums{\@tempb|\edfont@info}%
706       \global\let\@tempb=\undefined
707     \fi
708   \fi}
709

```

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 710 \newenvironment{pairs}{%}
711   \l@dpairingtrue
712   \l@dpagingfalse
713 }{%
714   \l@dpairingfalse
715 }

```

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.

```

716 \newenvironment{pages}{%
717   \let\oldchapter\chapter
718   \let\chapter\chapterinpages
719   \l@dpairingtrue
720   \l@dpagingtrue
721   \setlength{\Lcolwidth}{\textwidth}%
722   \setlength{\Rcolwidth}{\textwidth}%
723 }{%
724   \l@dpairingfalse
725   \l@dpagingfalse
726   \let\chapter\oldchapter
727 }
728 \newcommand{\chapterinpages}{\thispagestyle{plain}%
729   \global\@topnum\z@
730   \@afterindentfalse
731   \secdef\@chapter\@schapter}
732

```

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

733 \newif\ifinstanzaL
734 \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.
- ```

735 \newenvironment{Leftside}{%
736   \ledRcolfalse
737   \let\beginnumbering\beginnumbering\setcounter{pstartL}{1}
738   \let\pstart\pstartL
739   \let\thepstart\thepstartL
740   \let\pend\pendL
741   \let\memorydump\memorydumpL
742   \Leftsidehook
743   \let\oldstanza\stanza
744   \renewcommand{\stanza}{\oldstanza\global\instanzaLtrue}
745 }{
746   \let\stanza\oldstanza
747   \Leftsidehookend}

```
- `\Leftsidehook` Hooks into the start and end of the `Leftside` and `Rightside` environments. These are initially empty.
- `\Leftsidehookend`
- `\Rightsidehook` 748 `\newcommand*{\Leftsidehook}{}{}`
- `\Rightsidehookend` 749 `\newcommand*{\Leftsidehookend}{}{}`
- ```

750 \newcommand*{\Rightsidehook}{}{ }
751 \newcommand*{\Rightsidehookend}{}{ }
752

```
- 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.
- ```

753 \newenvironment{Rightside}{%
754   \ledRcoltrue
755   \let\beginnumbering\beginnumberingR
756   \let\endnumbering\endnumberingR
757   \let\pausenumbering\pausenumberingR
758   \let\resumenumbering\resumenumberingR
759   \let\memorydump\memorydumpR
760   \let\thepstart\thepstartR
761   \let\pstart\pstartR
762   \let\pend\pendR
763   \let\lineation\lineationR
764   \Rightsidehook
765   \let\oldstanza\stanza
766   \renewcommand{\stanza}{\oldstanza\global\instanzaRtrue}
767 }{%

```

```

768 \ledRcolfalse
769 \let\stanza\oldstanza
770 \Rightsidehookend
771 }
772

```

14 Paragraph decomposition and reassembly

In order to be able to count the lines of text and affix line numbers, we add an extra stage of processing for each paragraph. We send the paragraph into a box register, rather than straight onto the vertical list, and when the paragraph ends we slice the paragraph into its component lines; to each line we add any notes or line numbers, add a command to write to the line-list, and then at last send the line to the vertical list. This section contains all the code for this processing.

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

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

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

```

773 \newcount\num@linesR
774 \newbox\one@lineR
775 \newcount\par@lineR

```

`\pstartL` changesv1.12012/09/25Add `\labelpstarttrue` (from eledmac). changesv1.12012/10/01Correct
`\pstartR` `\pstartR` of 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.

```

776
777 \newcounter{pstartL}
778 \newcounter{pstartLold}
779 \renewcommand{\thepstartL}{\bfseries\@arabic\c@pstartL}. }

```

```

780 \newcounter{pstartR}
781 \newcounter{pstartRold}
782 \renewcommand{\thepstartR}{\bfseries\@arabic\c@pstartR}. }
783
784 \newcommand*{\pstartL}{
785   \if@nbreak
786     \let\@oldnbreak\@nbreaktrue
787   \else
788     \let\@oldnbreak\@nbreakfalse
789   \fi
790   \@nbreaktrue
791   \ifnumbering \else
792     \led@err@PstartNotNumbered
793     \beginnumbering
794   \fi
795   \ifnumberedpar@
796     \led@err@PstartInPstart
797   \pend
798 \fi

```

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

```

799 \ifpstart@rtedL\else
800   \setcounter{pstartLold}{\value{pstartL}}%
801   \list@clear{\inserts@list}%
802   \global\let\next@insert=\empty
803   \global\pstart@rtedLtrue
804 \fi
805 \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.

```

806 \global\advance\l@dnumpstartsL \@ne
807 \ifnum\l@dnumpstartsL>\l@dc@maxchunks
808   \led@err@TooManyPstarts
809   \global\l@dnumpstartsL=\l@dc@maxchunks
810 \fi
811 \global\setnamebox{\l@dLcolrawbox\the\l@dnumpstartsL}=\vbox\bgroup\ifautopar\else\ifnumb
812   \hsize=\Lcolwidth
813 \numberedpar@true
814 \iflabelpstart\protected@edef\@currentlabel
815   {\p@pstartL\thepstartL}\fi
816 }
817 \newcommand*{\pstartR}{
818   \if@nbreak
819     \let\@oldnbreak\@nbreaktrue
820   \else
821     \let\@oldnbreak\@nbreakfalse
822   \fi
823   \@nbreaktrue

```

```

824 \ifnumberingR \else
825   \led@err@PstartNotNumbered
826   \beginnumberingR
827 \fi
828 \ifnumberedpar@
829   \led@err@PstartInPstart
830   \pendR
831 \fi
832 \ifpst@rtedR\else
833   \setcounter{pstartRold}{\value{pstartR}}%
834   \list@clear{\inserts@listR}%
835   \global\let\next@insertR=\empty
836   \global\pst@rtedRtrue
837 \fi
838 \begingroup\normal@pars
839 \global\advance\l@dnumpstartsR \@ne
840 \ifnum\l@dnumpstartsR>\l@dc@maxchunks
841   \led@err@TooManyPstarts
842   \global\l@dnumpstartsR=\l@dc@maxchunks
843 \fi
844 \global\setnamebox{\l@dRcolrawbox\the\l@dnumpstartsR}=\vbox\bgroup\ifautopar\else\ifnumberpstart\if
845   \hsize=\Rcolwidth
846 \numberedpar@true
847 \iflabelpstart\protected@edef\@currentlabel
848   {\p@pstartR\thepstartR}\fi
849 }

```

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

```

850 \newcommand*{\pendL}{\ifnumbering \else
851   \led@err@PendNotNumbered
852 \fi
853 \ifnumberedpar@ \else
854   \led@err@PendNoPstart
855 \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.

```

856 \l@dzeropenalties
857 \endgraf\global\num@lines=\prevgraf\egroup
858 \global\par@line=0

```

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

```

859 \endgroup
860 \ignorespaces
861 \@oldnobreak
862 \ifnumberpstart
863   \addtocounter{pstartL}{1}

```

```
864 \fi}
865
```

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

```
866 \newcommand*{\pendR}{\ifnumberingR \else
867   \led@err@PendNotNumbered
868   \fi
869   \ifnumberedpar@ \else
870   \led@err@PendNoPstart
871   \fi
872   \l@dzeropenalties
873   \endgraf\global\num@linesR=\prevgraf\egroup
874   \global\par@lineR=0
875   \endgroup
876   \ignorespaces
877   \@oldnobreak
878   \ifnumberpstart
879   \addtocounter{pstartR}{1}
880   \fi
881 }
882
```

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 analogously for a line of right text.

```
883 \newbox\l@dleftbox
884 \newbox\l@drightbox
885
```

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

```
\countRline 886 \newcount\countLline
887   \countLline \z@
888 \newcount\countRline
889   \countRline \z@
890
```

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

```
\@donetotallinesL 891 \newcount\@donereallinesL
892 \newcount\@donetotallinesL
893 \newcount\@donereallinesR
894 \newcount\@donetotallinesR
895
```

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

```

896 \newcommand*{\do@lineL}{%
897   \advance\countLline \@ne
898   \ifvbox\namebox{\l@dLcolrawbox\the\l@dpscL}%
899   {\vbadness=10000
900    \splittopskip=\z@
901    \do@lineLhook
902    \l@emptyd@ta
903    \global\setbox\one@line=\vsplit\namebox{\l@dLcolrawbox\the\l@dpscL}
904                      to\baselineskip}%
905   \unvbox\one@line \global\setbox\one@line=\lastbox
906   \getline@numL
907   \ifnum\@lock>\@ne\inserthangingsymboltrue\else\inserthangingsymbolfalse\fi
908   \setbox\l@dleftbox
909   \hb@xt@ \Lcolwidth{%
910     \affixpstart@numL
911     \affixline@num
912     \l@dld@ta
913     \add@inserts
914     \affixside@note
915     \l@dlsn@te
916     {\ledllfill\hb@xt@ \wd\one@line{\inserthangingsymbolL\new@line\l@dunhbox@line{\one@line}}\correct
917     \l@drrsn@te
918   }}%
919   \add@penaltiesL
920   \global\advance\@donereallinesL\@ne
921   \global\advance\@donetotallinesL\@ne
922 \else
923   \setbox\l@dleftbox \hb@xt@ \Lcolwidth{\hspace*{\Lcolwidth}}%
924   \global\advance\@donetotallinesL\@ne
925 \fi}
926
927
```

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

```

\do@lineRhook 928 \newcommand*{\do@lineLhook}{}
               929 \newcommand*{\do@lineRhook}{}
               930
```

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

```

931 \newcommand*{\do@lineR}{%
932   \advance\countRline \@ne
933   \ifvbox\namebox{\l@dRcolrawbox\the\l@dpscR}%
934   {\vbadness=10000
935    \splittopskip=\z@
936    \do@lineRhook
937    \l@emptyd@ta
```

```

938 \global\setbox\one@lineR=\vsplit\namebox{1@dRcolrawbox\the\l@dpscR}
939 to\baselineskip}%
940 \unvbox\one@lineR \global\setbox\one@lineR=\lastbox
941 \getline@numR
942 \ifnum\@lockR>\@ne\inserthangingsymbolRtrue\else\inserthangingsymbolRfalse\fi
943 \setbox\l@drightbox
944 \hb@xt@ \Rcolwidth{%
945 \affixstart@numR
946 \affixline@numR
947 \l@dld@ta
948 \add@insertsR
949 \affixside@noteR
950 \l@dlsn@te
951 {\correcthangingsR\ledllfill\hb@xt@ \wd\one@lineR{\inserthangingsymbolR\new@lineR\l@du
952 \l@drsn@te
953 }}%
954 \add@penaltiesR
955 \global\advance\@donereallinesR\@ne
956 \global\advance\@donetotallinesR\@ne
957 \else
958 \setbox\l@drightbox \hb@xt@ \Rcolwidth{\hspace*{\Rcolwidth}}
959 \global\advance\@donetotallinesR\@ne
960 \fi}
961
962

```

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.

```

963 \newcommand*{\getline@numR}{%
964 \global\advance\absline@numR \@ne
965 \do@actionsR
966 \do@ballastR
967 \ifnumberline
968 \ifsublines@
969 \ifnum\sub@lockR<\tw@
970 \global\advance\subline@numR \@ne
971 \fi
972 \else
973 \ifnum\@lockR<\tw@
974 \global\advance\line@numR \@ne
975 \global\subline@numR \z@
976 \fi
977 \fi
978 \fi
979 }
980 \newcommand*{\getline@numL}{%
981 \global\advance\absline@num \@ne

```

```

982 \do@actions
983 \do@ballast
984 \ifnumberline
985 \ifsublines@
986   \ifnum\sub@lock<\tw@
987     \global\advance\subline@num \@ne
988   \fi
989 \else
990   \ifnum\@lock<\tw@
991     \global\advance\line@num \@ne
992     \global\subline@num \z@
993   \fi
994 \fi
995 \fi
996 }
997
998

```

`\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.

```

999 \newcommand*{\do@ballastR}{\global\ballast@count=\z@
1000 \begingroup
1001   \advance\absline@numR \@ne
1002   \ifnum\next@actionlineR=\absline@numR
1003     \ifnum\next@actionR>-1001
1004       \global\advance\ballast@count by -\c@ballast
1005     \fi
1006   \fi
1007 \endgroup}

```

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

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

```

1008 \newcommand*{\do@actions@fixedcodeR}{%
1009   \ifcase\@l@dttempcnta%
1010   \or% % 1001
1011     \global\sublines@true
1012   \or% % 1002
1013     \global\sublines@false
1014   \or% % 1003
1015     \global\@lockR=\@ne
1016   \or% % 1004
1017     \ifnum\@lockR=\tw@
1018       \global\@lockR=\thr@@
1019     \else
1020       \global\@lockR=\z@
1021     \fi

```

```

1022 \or% % 1005
1023 \global\sub@lockR=\@ne
1024 \or% % 1006
1025 \ifnum\sub@lockR=\tw@
1026 \global\sub@lockR=\thr@@
1027 \else
1028 \global\sub@lockR=\z@
1029 \fi
1030 \or% % 1007
1031 \l@dskipnumbertrue
1032 \else
1033 \led@warn@BadAction
1034 \fi}
1035
1036
1037 \newcommand*{\do@actionsR}{%
1038 \global\let\do@actions@nextR=\relax
1039 \@l@dttempcntb=\absline@numR
1040 \ifnum\@l@dttempcntb<\next@actionlineR\else
1041 \ifnum\next@actionR>-1001\relax
1042 \global\page@numR=\next@actionR
1043 \ifbypage@R
1044 \global\line@numR \z@ \global\subline@numR \z@
1045 \fi
1046 \else
1047 \ifnum\next@actionR<-4999\relax % 9/05 added relax here
1048 \@l@dttempcnta=-\next@actionR
1049 \advance\@l@dttempcnta by -5001\relax
1050 \ifsublines@
1051 \global\subline@numR=\@l@dttempcnta
1052 \else
1053 \global\line@numR=\@l@dttempcnta
1054 \fi
1055 \else
1056 \@l@dttempcnta=-\next@actionR
1057 \advance\@l@dttempcnta by -1000\relax
1058 \do@actions@fixedcodeR
1059 \fi
1060 \fi
1061 \ifx\actionlines@listR\empty
1062 \gdef\next@actionlineR{1000000}%
1063 \else
1064 \glp\actionlines@listR\to\next@actionlineR
1065 \glp\actions@listR\to\next@actionR
1066 \global\let\do@actions@nextR=\do@actionsR
1067 \fi
1068 \fi
1069 \do@actions@nextR}
1070

```

14.4 Line number printing

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

```

\ch@cksub@l@ckR 1071
\ch@ck@l@ckR 1072 \providecommand*\l@dcalcnm}[3]{%
\fx@l@cksR 1073 \ifnum #1 > #2\relax
\affixline@numR 1074 \@l@dtmpcnta = #1\relax
1075 \advance\@l@dtmpcnta by -#2\relax
1076 \divide\@l@dtmpcnta by #3\relax
1077 \multiply\@l@dtmpcnta by #3\relax
1078 \advance\@l@dtmpcnta by #2\relax
1079 \else
1080 \@l@dtmpcnta=#2\relax
1081 \fi}
1082
1083 \newcommand*\ch@cksub@l@ckR}{%
1084 \ifcase\sub@lockR
1085 \or
1086 \ifnum\sublock@disp=\@ne
1087 \@l@dtmpcntb \z@ \@l@dtmpcnta \@ne
1088 \fi
1089 \or
1090 \ifnum\sublock@disp=\tw@
1091 \else
1092 \@l@dtmpcntb \z@ \@l@dtmpcnta \@ne
1093 \fi
1094 \or
1095 \ifnum\sublock@disp=\z@
1096 \@l@dtmpcntb \z@ \@l@dtmpcnta \@ne
1097 \fi
1098 \fi}
1099
1100 \newcommand*\ch@ck@l@ckR}{%
1101 \ifcase\@lockR
1102 \or
1103 \ifnum\lock@disp=\@ne
1104 \@l@dtmpcntb \z@ \@l@dtmpcnta \@ne
1105 \fi
1106 \or
1107 \ifnum\lock@disp=\tw@
1108 \else
1109 \@l@dtmpcntb \z@ \@l@dtmpcnta \@ne
1110 \fi
1111 \or
1112 \ifnum\lock@disp=\z@
1113 \@l@dtmpcntb \z@ \@l@dtmpcnta \@ne
1114 \fi
1115 \fi}
1116
1117 \newcommand*\fx@l@cksR}{%

```

```

1118 \ifcase\@lockR
1119 \or
1120 \global\@lockR \tw@
1121 \or \or
1122 \global\@lockR \z@
1123 \fi
1124 \ifcase\sub@lockR
1125 \or
1126 \global\sub@lockR \tw@
1127 \or \or
1128 \global\sub@lockR \z@
1129 \fi}
1130
1131
1132 \newcommand*{\affixline@numR}{%
1133 \ifnumberline
1134 \ifl@dskipnumber
1135 \global\l@dskipnumberfalse
1136 \else
1137 \ifsublines@
1138 \@l@tempcntb=\subline@numR
1139 \l@dcalcnnum{\subline@numR}{\c@firstsublinenumR}{\c@sublinenumincrementR}%
1140 \ch@cksub@lockR
1141 \else
1142 \@l@tempcntb=\line@numR
1143 \ifx\linenumberlist\empty
1144 \l@dcalcnnum{\line@numR}{\c@firstlinenumR}{\c@linenumincrementR}%
1145 \else
1146 \@l@tempcnta=\line@numR
1147 \edef\rem@inder{,\linenumberlist,\number\line@numR,}%
1148 \edef\sc@n@list{\def\noexpand\sc@n@list
1149 ###1,\number\@l@tempcnta,###2|{\def\noexpand\rem@inder{###2}}}%
1150 \sc@n@list\expandafter\sc@n@list\rem@inder|}%
1151 \ifx\rem@inder\empty\advance\@l@tempcnta\@ne\fi
1152 \fi
1153 \ch@ck@l@ckR
1154 \fi
1155 \ifnum\@l@tempcnta=\@l@tempcntb
1156 \iftwocolumn
1157 \if@firstcolumn
1158 \gdef\l@dld@ta{\llap{\leftlinenumR}}}%
1159 \else
1160 \gdef\l@drd@ta{\rlap{\rightlinenumR}}}%
1161 \fi
1162 \else
1163 \@l@tempcntb=\line@marginR
1164 \ifnum\@l@tempcntb>\@ne
1165 \advance\@l@tempcntb by\page@numR
1166 \fi
1167 \ifodd\@l@tempcntb

```

```

1168      \gdef\l@drd@ta{\rlap{{\rightlinenumR}}}%
1169      \else
1170      \gdef\l@dld@ta{\llap{{\leftlinenumR}}}%
1171      \fi
1172  \fi
1173  \fi
1174  \f@x@l@cksR
1175 \fi
1176 \fi}

```

14.5 Pstart number printing in side

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

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

```

\affixpstart@numL
\affixpstart@numR 1177
  \leftpstartnumR 1178 \newcommand*{\affixpstart@numL}{%
\rightpstartnumR 1179 \ifsidepstartnum
\leftpstartnumL 1180 \if@twocolumn
\rightpstartnumL 1181   \if@firstcolumn
  \ifpstartnumR 1182     \gdef\l@dld@ta{\llap{{\leftpstartnumL}}}%
    1183     \else
    1184     \gdef\l@drd@ta{\rlap{{\rightpstartnumL}}}%
    1185     \fi
    1186   \else
    1187     \l@dttempcntb=\line@margin
    1188     \ifnum\l@dttempcntb>\@ne
    1189       \advance\l@dttempcntb \page@num
    1190     \fi
    1191     \ifodd\l@dttempcntb
    1192       \gdef\l@drd@ta{\rlap{{\rightpstartnumL}}}%
    1193     \else
    1194       \gdef\l@dld@ta{\llap{{\leftpstartnumL}}}%
    1195     \fi
    1196   \fi
    1197 \fi
    1198 }
    1199 \newcommand*{\affixpstart@numR}{%
    1200 \ifsidepstartnum
    1201 \if@twocolumn
    1202   \if@firstcolumn
    1203     \gdef\l@dld@ta{\llap{{\leftpstartnumR}}}%
    1204   \else
    1205     \gdef\l@drd@ta{\rlap{{\rightpstartnumR}}}%

```

```

1206     \fi
1207   \else
1208     \@l@tempcntb=\line@marginR
1209     \ifnum\@l@tempcntb>\@ne
1210       \advance\@l@tempcntb \page@numR
1211     \fi
1212     \ifodd\@l@tempcntb
1213       \gdef\l@drd@ta{\rlap{\rightstartnumR}}}%
1214     \else
1215       \gdef\l@dld@ta{\llap{\leftstartnumR}}}%
1216     \fi
1217   \fi
1218 \fi
1219 }
1220
1221 \newcommand*\leftstartnumL{%
1222 \ifpstartnum
1223 \thepstartL
1224 \kern\linenumsep\global\pstartnumfalse\fi
1225 }
1226 \newcommand*\rightstartnumL{%
1227 \ifpstartnum\kern\linenumsep
1228 \thepstartL
1229 \global\pstartnumfalse\fi
1230 }
1231 \newif\ifpstartnumR
1232 \pstartnumRtrue
1233 \newcommand*\leftstartnumR{%
1234 \ifpstartnumR
1235 \thepstartR
1236 \kern\linenumsep\global\pstartnumRfalse\fi
1237 }
1238 \newcommand*\rightstartnumR{%
1239 \ifpstartnumR\kern\linenumsep
1240 \thepstartR
1241 \global\pstartnumRfalse\fi
1242 }

```

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.

```
1243 \list@create{\inserts@listR}
```

`\add@insertsR` The right text version.

```

\add@inserts@nextR 1244 \newcommand*\add@insertsR{%
1245   \global\let\add@inserts@nextR=\relax
1246   \ifx\inserts@listR\empty \else
1247     \ifx\next@insertR\empty

```

```

1248     \ifx\insertlines@listR\empty
1249         \global\noteschanged@true
1250         \gdef\next@insertR{100000}%
1251     \else
1252         \gl@p\insertlines@listR\to\next@insertR
1253     \fi
1254 \fi
1255 \ifnum\next@insertR=\absline@numR
1256     \gl@p\inserts@listR\to\@insertR
1257     \@insertR
1258     \global\let\@insertR=\undefined
1259     \global\let\next@insertR=\empty
1260     \global\let\add@inserts@nextR=\add@insertsR
1261 \fi
1262 \fi
1263 \add@inserts@nextR}
1264

```

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

```

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

```

1265 \newcommand*{\add@penaltiesL}{%
1266 \newcommand*{\add@penaltiesR}{%
1267

```

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.

```

1268 \newcommand*{\flush@notesR}{%
1269 \xloop
1270 \ifx\inserts@listR\empty \else
1271 \glp\inserts@listR\to\@insertR
1272 \@insertR
1273 \global\let\@insertR=\undefined
1274 \repeat}
1275

```

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@dt@nums`, in the form described on page ??: the starting page, line, and sub-line numbers, followed by the ending page, line, and sub-line numbers, and then the font specifier for the lemma.

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

Just a reminder of the arguments:

```

\printlinesR #1 | #2 | #3 | #4 | #5 | #6 | #7
\printlinesR start-page | line | subline | end-page | line | subline | font
1276 \def\printlinesR#1|#2|#3|#4|#5|#6|#7|{\begingroup
1277 \setprintlines{#1}{#2}{#3}{#4}{#5}{#6}%
1278 \ifl@dt@pnum #1\fullstop\fi

```

```

1279 \ifledplinenum \linenumr@p{#2}\Rlineflag\else \symlinenum\fi
1280 \ifl@d@ssub \fullstop \sublinenumr@p{#3}\fi
1281 \ifl@d@dash \endashchar\fi
1282 \ifl@d@pnum #4\fullstop\fi
1283 \ifl@d@elin \linenumr@p{#5}\Rlineflag\fi
1284 \ifl@d@esl \ifl@d@elin \fullstop\fi \sublinenumr@p{#6}\fi
1285 \endgroup}
1286
1287 \let\ledsavedprintlines\printlines
1288

```

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.

```

1289 \list@create{\labelref@listR}
1290

```

`\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.

```

1291 \renewcommand*{\edlabel}[1]{\@bsphack
1292 \ifledRcol
1293 \write\linenum@outR{\string\@lab}%
1294 \ifx\labelref@listR\empty
1295 \xdef\label@refs{\zz@@@}%
1296 \else
1297 \glp\labelref@listR\to\label@refs
1298 \fi
1299 \ifvmode
1300 \advancelabel@refs
1301 \fi
1302 \protected@write\@auxout{%
1303 {\string\l@dmake@labelsR\space\thepage|\label@refs|{#1}}%
1304 \else
1305 \write\linenum@out{\string\@lab}%
1306 \ifx\labelref@list\empty
1307 \xdef\label@refs{\zz@@@}%
1308 \else
1309 \glp\labelref@list\to\label@refs
1310 \fi
1311 \ifvmode
1312 \advancelabel@refs
1313 \fi
1314 \protected@write\@auxout{%
1315 {\string\l@dmake@labels\space\thepage|\label@refs|{#1}}%

```

```

1316 \fi
1317 \@esphack}
1318

```

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

```

1319 \def\l@dmake@labelsR#1|#2|#3|#4{%
1320 \expandafter\ifx\csname the@label#4\endcsname \relax\else
1321 \led@warn@DuplicateLabel{#4}%
1322 \fi
1323 \expandafter\gdef\csname the@label#4\endcsname{#1|#2\Rlineflag|#3}%
1324 \ignorespaces}
1325 \AtBeginDocument{%
1326 \def\l@dmake@labelsR#1|#2|#3|#4{%
1327 }
1328

```

`\@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.

```

1329 \renewcommand*{\@lab}{%
1330 \ifledRcol
1331 \xright@appenditem{\linenumr@p{\line@numR}}{|%
1332 \ifsublines@ \sublinenumr@p{\subline@numR}\else 0\fi}%
1333 \to\labelref@listR
1334 \else
1335 \xright@appenditem{\linenumr@p{\line@num}}{|%
1336 \ifsublines@ \sublinenumr@p{\subline@num}\else 0\fi}%
1337 \to\labelref@list
1338 \fi}
1339

```

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 1340 \newcount\sidenote@marginR
1341 \renewcommand*{\sidenotemargin}[1]{%
1342 \l@dgetsidenote@margin{#1}%
1343 \ifnum\@l@tempcntb>\m@ne
1344 \ifledRcol
1345 \global\sidenote@marginR=\@l@tempcntb
1346 \else
1347 \global\sidenote@margin=\@l@tempcntb
1348 \fi
1349 \fi}}

```

```

1350 \sidenotemargin{right}
1351 \global\sidenote@margin=\@ne
1352

```

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

```

\l@dcsnote 1353 \renewcommand*{\l@dlsnote}[1]{%
1354   \ifnumberedpar@
1355     \ifledRcol%
1356       \xright@appenditem{\noexpand\l@dlsnote{#1}}%
1357         \to\inserts@listR
1358     \else%
1359       \xright@appenditem{\noexpand\l@dlsnote{#1}}%
1360         \to\inserts@list
1361       \global\advance\insert@count \@ne%
1362     \fi
1363   \fi\ignorespaces}
1364 \renewcommand*{\l@drsnote}[1]{%
1365   \ifnumberedpar@
1366     \ifledRcol%
1367       \xright@appenditem{\noexpand\l@drsnote{#1}}%
1368         \to\inserts@listR
1369       \global\advance\insert@countR \@ne%
1370     \else%
1371       \xright@appenditem{\noexpand\l@drsnote{#1}}%
1372         \to\inserts@list
1373       \global\advance\insert@count \@ne%
1374     \fi
1375   \fi\ignorespaces}
1376 \renewcommand*{\l@dcsnote}[1]{%
1377   \ifnumberedpar@
1378     \ifledRcol%
1379       \xright@appenditem{\noexpand\l@dcsnote{#1}}%
1380         \to\inserts@listR
1381       \global\advance\insert@countR \@ne%
1382     \else%
1383       \xright@appenditem{\noexpand\l@dcsnote{#1}}%
1384         \to\inserts@list
1385       \global\advance\insert@count \@ne%
1386     \fi
1387   \fi\ignorespaces}
1388

```

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

```

1389 \newcommand*{\affixside@noteR}{%
1390   \def\sidenotecontent@{}%
1391   \numdef{\itemcount@}{0}%
1392   \renewcommand{\do}[1]{%
1393     \ifnumequal{\itemcount@}{0}%
1394       {%

```

```

1395         \appto\sidenotecontent@{##1}}% Not print not separator before the 1st note
1396         {\appto\sidenotecontent@{\sidenotessep ##1}}%
1397         }%
1398         \numdef{\itemcount@}{\itemcount@+1}%
1399     }%
1400     \dolistloop{\l@dcnotetext}%
1401     \ifnumgreater{\itemcount@}{1}{\eledmac@warning{\itemcount@\space sidenotes on line \tl
1402 \gdef\@templ@d}{%
1403 \ifx\@templ@d\l@dcnotetext \else%
1404     \if@twocolumn%
1405         \if@firstcolumn%
1406             \setl@dlp@rbox{\sidenotecontent@}%
1407         \else%
1408             \setl@drp@rbox{\sidenotecontent@}%
1409         \fi%
1410     \else%
1411         \@l@tempcntb=\sidenote@marginR%
1412         \ifnum\@l@tempcntb>\@ne%
1413             \advance\@l@tempcntb by\page@num%
1414         \fi%
1415         \ifodd\@l@tempcntb%
1416             \setl@drp@rbox{\sidenotecontent@t}%
1417         \else%
1418             \setl@dlp@rbox{\sidenotecontent@}%
1419         \fi%
1420     \fi%
1421 \fi}
1422

```

18 Familiar footnotes

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

```

1423 \renewcommand{\l@dbfnote}[1]{%
1424     \ifnumberedpar@
1425         \ifledRcol%
1426             \xright@appenditem{\noexpand\vl@dbfnote{##1}}{\@thefnmark}}%
1427             \to\inserts@listR
1428         \global\advance\insert@countR \@ne%
1429     \else%
1430         \xright@appenditem{\noexpand\vl@dbfnote{##1}}{\@thefnmark}}%
1431         \to\inserts@list
1432         \global\advance\insert@count \@ne%
1433     \fi
1434 \fi\ignorespaces}
1435

```

`\normalbfnoteX`

```

1436 \renewcommand{\normalbfnoteX}[2]{%
1437   \ifnumberedpar@
1438     \ifledRcol%
1439       \xright@appenditem{\noexpand\vbfnoteX{#1}{#2}{\@nameuse{thefootnote#1}}}%
1440       \to\inserts@listR
1441       \global\advance\insert@countR \@ne%
1442     \else%
1443       \xright@appenditem{\noexpand\vbfnoteX{#1}{#2}{\@nameuse{thefootnote#1}}}%
1444       \to\inserts@list
1445       \global\advance\insert@count \@ne%
1446     \fi
1447 \fi\ignorespaces}
1448

```

19 Verse

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

```

\inserthangingsymbolL
\inserthangingsymbolR 1449 \newif\ifinserthangingsymbolR
1450 \newcommand{\inserthangingsymbolL}{%
1451   \ifinserthangingsymbol%
1452     \ifinstanzaL%
1453       \hfill\hangingsymbol%
1454     \fi%
1455 \fi}
1456 \newcommand{\inserthangingsymbolR}{%
1457   \ifinserthangingsymbolR%
1458     \ifinstanzaR%
1459       \hfill\hangingsymbol%
1460     \fi%
1461 \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 1462 \newcommand{\correcthangingL}{%
1463   \ifl@dpaging\else%
1464     \ifinstanzaL%
1465       \ifinserthangingsymbol%
1466         \hskip \@ifundefined{sza@00}{0}{\expandafter%
1467           \noexpand\csname sza@00\endcsname}\stanzaindentbase%
1468       \fi%
1469     \fi%
1470 \fi}

```

```

1471
1472 \newcommand{\correcthangingR}{%
1473 \ifl@dpaging\else%
1474   \ifinstanzaR%
1475     \ifinserthangingsymbolR%
1476       \hskip \@ifundefined{sza@0@}{0}{\expandafter%
1477         \noexpand\csname sza@0@\endcsname}\stanzaindentbase%
1478     \fi%
1479   \fi%
1480 \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.

```

1481 \chardef\next=\catcode'\&
1482 \catcode'\&=\active
1483

```

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

```

1484 \newenvironment{astanza}{%
1485   \startstanzahook
1486   \catcode'\&\active
1487   \global\stanza@count\@ne
1488   \ifnum\usernamecount{sza@0@}=\z@
1489     \let\stanza@hang\relax
1490     \let\endlock\relax
1491   \else
1492   %% \interlinepenalty\@M % this screws things up, but I don't know why
1493   \rightskip\z@ plus 1fil\relax
1494   \fi
1495   \ifnum\usernamecount{szp@0@}=\z@
1496     \let\sza@penalty\relax
1497   \fi
1498   \def&{%
1499     \endlock\mbox{}}%
1500     \sza@penalty
1501   \global\advance\stanza@count\@ne
1502   \@astanza@line}%
1503   \def\&{%
1504     \endlock\mbox{}}
1505   \pend
1506   \endstanzaextra}%
1507   \pstart
1508   \@astanza@line
1509 }{}
1510

```

\@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.

```

1511 \newcommand*{\@astanza@line}{%
1512   \parindent=\csname sza@\number\stanza@count @\endcsname\stanzaindentbase
1513   \par
1514   \stanza@hang%\mbox{}%
1515   \ignorespaces}
1516

```

Lastly reset the modified category codes.

```

1517 \catcode'\&=\next
1518

```

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 1519 \providecommand*{\newnamebox}[1]{%
\unvnamebox 1520   \expandafter\newbox\csname #1\endcsname}
\namebox 1521 \providecommand*{\setnamebox}[1]{%
  1522   \expandafter\setbox\csname #1\endcsname}
  1523 \providecommand*{\unhnamebox}[1]{%
  1524   \expandafter\unhbox\csname #1\endcsname}
  1525 \providecommand*{\unvnamebox}[1]{%
  1526   \expandafter\unvbox\csname #1\endcsname}
  1527 \providecommand*{\namebox}[1]{%
  1528   \csname #1\endcsname}
  1529

```

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

```

\usenamecount 1530 \providecommand*{\newnamecount}[1]{%
  1531   \expandafter\newcount\csname #1\endcsname}
  1532 \providecommand*{\usenamecount}[1]{%
  1533   \csname #1\endcsname}
  1534

```

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.

```

1535 \newcount\l@dc@maxchunks
1536 \newcommand{\maxchunks}[1]{\l@dc@maxchunks=#1}
1537 \maxchunks{5120}
1538

```

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

```

\l@dnumpstartsR 1539 \newcount\l@dnumpstartsR
1540

```

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

```

\l@psclR 1541 \newcount\l@dpscl
1542 \newcount\l@dpsclR
1543

```

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

```

1544 \newcommand*{\l@dsetuprawboxes}{%
1545 \l@l@tempcntb=\l@dc@maxchunks
1546 \loop\ifnum\l@l@tempcntb>\z@
1547 \newnamebox{\l@dLcolrawbox\the\l@l@tempcntb}
1548 \newnamebox{\l@dRcolrawbox\the\l@l@tempcntb}
1549 \advance\l@l@tempcntb \m@ne
1550 \repeat}
1551

```

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

```

1552 \newcommand*{\l@dsetupmaxlinecounts}{%
1553 \l@l@tempcntb=\l@dc@maxchunks
1554 \loop\ifnum\l@l@tempcntb>\z@
1555 \newnamecount{\l@dmaxlinesinpar\the\l@l@tempcntb}
1556 \advance\l@l@tempcntb \m@ne
1557 \repeat}
1558 \newcommand*{\l@dzeromaxlinecounts}{%
1559 \begingroup
1560 \l@l@tempcntb=\l@dc@maxchunks
1561 \loop\ifnum\l@l@tempcntb>\z@
1562 \global\usenamecount{\l@dmaxlinesinpar\the\l@l@tempcntb}=\z@
1563 \advance\l@l@tempcntb \m@ne
1564 \repeat
1565 \endgroup}
1566

```

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

```

1567 \AtBeginDocument{%
1568 \l@dsetuprawboxes

```

```

1569 \l@dssetupmaxlinecounts
1570 \l@dzeromaxlinecounts
1571 \l@dnumpstartsL=\z@
1572 \l@dnumpstartsR=\z@
1573 \l@dpscL=\z@
1574 \l@dpscR=\z@}
1575

```

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 1576 \newif\ifl@dusedbabel
\l@dusedbabeltrue 1577 \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 1578 \newif\ifl@dsamelang
1579 \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.
1580 \newcommand*{\l@dchecklang}{%
1581 \l@dsamelangfalse
1582 \edef\@tempa{\theledlanguageL}\edef\@tempb{\theledlanguageR}%
1583 \ifx\@tempa\@tempb
1584 \l@dsamelangtrue
1585 \fi}
1586

\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 \.
1587 \newcommand*{\l@dbbl@set@language}[1]{%
1588 \edef\language{#1}%

```

```

1589 \select@language{\language}%
1590 \if@files
1591   \protected@write\auxout{}\string\select@language{\language}%
1592   \addtocontents{toc}{\string\select@language{\language}%
1593   \addtocontents{lof}{\string\select@language{\language}%
1594   \addtocontents{lot}{\string\select@language{\language}%
1595   \fi}
1596

```

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
1597 \providecommand{\selectlanguage}[1]{
1598 \newcommand*\l@duselanguage}[1]{
1599 \gdef\theledlanguageL{
1600 \gdef\theledlanguageR{
1601

```

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

```

1602 \AtBeginDocument{%
1603   \ifundefined{xpg@main@language}{%
1604     \ifundefined{bbl@main@language}{%

```

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

```

1605   \l@dusedbabelfalse
1606   \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.

```

1607   \l@dusedbabeltrue
1608   \let\l@doldselectlanguage\selectlanguage
1609   \let\l@doldbbl@set@language\bbl@set@language
1610   \let\bbl@set@language\l@dbbl@set@language
1611   \renewcommand{\selectlanguage}[1]{%
1612     \l@doldselectlanguage{#1}%
1613     \ifledRcol \gdef\theledlanguageR{#1}%
1614     \else      \gdef\theledlanguageL{#1}%
1615     \fi}

```

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

```

1616   \renewcommand*\l@duselanguage}[1]{%
1617     \l@doldselectlanguage{#1}%

```

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

```

1618   \gdef\theledlanguageL{\bbl@main@language}%

```

```

1619 \gdef\theledlanguageR{\bbl@main@language}%
1620 }%
1621 }

  If on Polyglossia
1622 { \apptocmd{\xpg@set@language}{%
1623   \ifledRcol \gdef\theledlanguageR{#1}%
1624   \else      \gdef\theledlanguageL{#1}%
1625   \fi}%
1626   \let\l@duselanguage\xpg@set@language
1627   \gdef\theledlanguageL{\xpg@main@language}%
1628   \gdef\theledlanguageR{\xpg@main@language}%
1629 % \end{macrocode}
1630 % That's it.
1631 % \begin{macrocode}
1632 }}

```

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.

```

1633 \newcommand*{\Columns}{%
1634   \setcounter{pstartL}{\value{pstartLold}}
1635   \setcounter{pstartR}{\value{pstartRold}}
1636   \ifnum\l@dnumpstartsL=\l@dnumpstartsR\else
1637     \led@err@BadLeftRightPstarts{\the\l@dnumpstartsL}{\the\l@dnumpstartsR}%
1638   \fi

```

Start a group and zero counters, etc.

```

1639 \begingroup
1640   \l@dzeropenalties
1641   \endgraf\global\num@lines=\prevgraf
1642   \global\num@linesR=\prevgraf
1643   \global\par@line=\z@
1644   \global\par@lineR=\z@
1645   \global\l@dpscL=\z@
1646   \global\l@dpscR=\z@

```

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

```

1647   \check@pstarts
1648   \loop\if@pstarts
1649     \global\pstartnumtrue
1650     \global\pstartnumRtrue

```

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

```

1651   \global\advance\l@dpscL \@ne
1652   \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

```
1653      \checkraw@text
1654      \l@dchecklang
1655 {      \loop\ifaraw@text
```

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

```
1656      \ifl@dsamelang
1657      \do@lineL
1658      \do@lineR
1659      \else
1660      \l@duselanguage{\theledlanguageL}%
1661      \do@lineL
1662      \l@duselanguage{\theledlanguageR}%
1663      \do@lineR
1664      \fi
1665      \hb@xt@ \hsize{%
1666      \hfill \unhbox\l@dleftbox
1667      \hfill \columnseparator \hfill
1668      \unhbox\l@drightbox
1669      }%
1670      \checkraw@text
1671      \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.

```
1672      \@writelinesinparL
1673      \@writelinesinparR
1674      \check@pstarts
1675      \ifbypstart@
1676      \write\linenum@out{\string\@set[1]}
1677      \resetprevline@
1678      \fi
1679      \ifbypstart@R
1680      \write\linenum@outR{\string\@set[1]}
1681      \resetprevline@
1682      \fi
1683      \addtocounter{pstartL}{1}
1684      \addtocounter{pstartR}{1}
1685      \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.

```
1686      \flush@notes
1687      \flush@notesR
1688      \endgroup
1689      \global\l@dpscL=\z@
1690      \global\l@dpscR=\z@
```

```

1691 \global\l@dnumstartsL=\z@
1692 \global\l@dnumstartsR=\z@
1693 \ignorespaces
1694 \global\instanzaLfalse
1695 \global\instanzaRfalse}
1696

```

`\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).

```

1697 \newcommand*{\columnseparator}{%
1698   \smash{\rule[-0.2\baselineskip]{\columnrulewidth}{1.05\baselineskip}}
1699 \newdimen\columnrulewidth
1700 \columnrulewidth=\z@
1701

```

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

```

\@pstartstrue 1702 \newif\if@pstarts
\@pstartsfalse 1703 \newcommand*{\check@pstarts}{%
\check@pstarts 1704   \@pstartsfalse
                  1705   \ifnum\l@dnumstartsL>\l@dpscl
                  1706     \@pstartstrue
                  1707   \else
                  1708     \ifnum\l@dnumstartsR>\l@dpscl
                  1709       \@pstartstrue
                  1710     \fi
                  1711   \fi
                  1712 }
                  1713

```

`\ifaraw@text` `\checkraw@text` checks whether the current Left or Right box is void or not. If `\araw@texttrue` one or other is not void it sets `\araw@texttrue`, otherwise both are void and it `\araw@textfalse` sets `\araw@textfalse`.

```

\checkraw@text 1714 \newif\ifaraw@text
                  1715   \araw@textfalse
                  1716 \newcommand*{\checkraw@text}{%
                  1717   \araw@textfalse
                  1718   \ifvbox\namebox{\l@dLcolrawbox\the\l@dpscl}
                  1719     \araw@texttrue
                  1720   \else
                  1721     \ifvbox\namebox{\l@dRcolrawbox\the\l@dpscl}
                  1722       \araw@texttrue
                  1723     \fi
                  1724   \fi
                  1725 }
                  1726

```

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

```

1727 \newcommand*{\@writelinesinparL}{%
1728   \edef\next{%
1729     \write\linenum@out{\string\@pend[\the\@donereallinesL]}}%
1730   \next
1731   \global\@donereallinesL \z@}
1732 \newcommand*{\@writelinesinparR}{%
1733   \edef\next{%
1734     \write\linenum@outR{\string\@pendR[\the\@donereallinesR]}}%
1735   \next
1736   \global\@donereallinesR \z@}
1737

```

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` 1738 \newcount\numpagelinesL
1739 \newcount\numpagelinesR
1740 \newcount\l@dminpagelines
1741

`\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.

```

1742 \newcommand*{\Pages}{%
1743   \setcounter{pstartL}{\value{pstartLold}}
1744   \setcounter{pstartR}{\value{pstartRold}}
1745   \typeout{}
1746   \typeout{***** PAGES *****}
1747   \ifnum\l@dnumpstartsL=\l@dnumpstartsR\else
1748     \led@err@BadLeftRightPstarts{\the\l@dnumpstartsL}{\the\l@dnumpstartsR}%
1749   \fi

```

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

```

1750 \clearto\l@devenpage
1751 \begingroup
1752   \l@dzeropenalties
1753   \endgraf\global\num@lines=\prevgraf
1754   \global\num@linesR=\prevgraf
1755   \global\par@line=\z@
1756   \global\par@lineR=\z@
1757   \global\l@dpscL=\z@
1758   \global\l@dpscR=\z@
1759   \writtenlinesLfalse
1760   \writtenlinesRfalse

```

Check if there are chunks to be processed.

```
1761 \check@pstarts
1762 \loop\if@pstarts
```

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

```
1763 \global\advance\l@dpscL \@ne
1764 \global\advance\l@dpscR \@ne
```

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

```
1765 \getlinesfromparlistL
1766 \getlinesfromparlistR
1767 \l@dcalc@maxoftwo{\@cs@linesinparL}{\@cs@linesinparR}%
1768 {\usernamecount{l@dmmaxlinesinpar\the\l@dpscL}}%
1769 \check@pstarts
1770 \repeat
```

Zero the counts again, ready for the next bit.

```
1771 \global\l@dpscL=\z@
1772 \global\l@dpscR=\z@
```

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

```
1773 \getlinesfrompagelistL
1774 \getlinesfrompagelistR
1775 \l@dcalc@minoftwo{\@cs@linesonpageL}{\@cs@linesonpageR}%
1776 {\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.

```
1777 \check@pstarts
1778 \if@pstarts
```

Increment the chunk counts to get the first pair.

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

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

```
1781 \global\@donereallinesL=\z@
1782 \global\@donetotallinesL=\z@
1783 \global\@donereallinesR=\z@
1784 \global\@donetotallinesR=\z@
```

Start a loop over the boxes (chunks).

```
1785 \checkraw@text
1786 % \begingroup
1787 { \loop\ifaraw@text
```

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

```

1788      \checkpageL
1789      \l@duselanguage{\theledlanguageL}%
1790      %%
1791      \beginngroup
1792      \loop\ifl@dsamepage

```

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

```

1793      \do@lineL
1794      \advance\numpagelinesL \@ne
1795      \ifshiftedpstarts
1796          \ifdim\ht\l@dleftbox>0pt\hb@xt@ \hsize{\ledstrutL\unhbox\l@dleftbox}%
1797      \else
1798          \hb@xt@ \hsize{\ledstrutL\unhbox\l@dleftbox}%
1799      \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.

```

1800
1801      \get@nextboxL
1802      \checkpageL
1803      \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.

```

1804      \ifl@dpagfull
1805      \@writelinesonpageL{\the\numpagelinesL}%
1806      \else
1807      \@writelinesonpageL{1000}%
1808      \fi

```

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

```

1809      \numpagelinesL \z@
1810      \clearl@dleftpage }%

```

Now do the same for the right text.

```

1811      \checkpageR
1812      \l@duselanguage{\theledlanguageR}%
1813      {
1814          \loop\ifl@dsamepage
1815          \do@lineR
1816          \advance\numpagelinesR \@ne
1817          \ifshiftedpstarts
1818              \ifdim\ht\l@drighbox>0pt\hb@xt@ \hsize{\ledstrutR\unhbox\l@drighbox}%
1819          \else
1820              \hb@xt@ \hsize{\ledstrutR\unhbox\l@drighbox}%
1821          \fi
1822          \get@nextboxR

```

```

1822         \checkpageR
1823     \repeat
1824     \ifl@dpagfull
1825         \@writelinesonpageR{\the\numpagelinesR}%
1826     \else
1827         \@writelinesonpageR{1000}%
1828     \fi
1829     \numpagelinesR=\z@

```

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

```
1830     \clearl@drighthpage}
```

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.

```

1831     \checkraw@text
1832     \ifaraw@text
1833         \getlinesfrompagelistL
1834         \getlinesfrompagelistR
1835         \l@dcalc@minoftwo{\@cs@linesonpageL}{\@cs@linesonpageR}%
1836             {\l@dminpagelines}%
1837     \fi
1838     \repeat}

```

We have now output the text from all the chunks.

```
1839     \fi
```

Make sure that there are no inserts hanging around.

```

1840     \flush@notes
1841     \flush@notesR
1842     \endgroup

```

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

```

1843     \global\l@dpscL=\z@
1844     \global\l@dpscR=\z@
1845     \global\l@dnumpestartsL=\z@
1846     \global\l@dnumpestartsR=\z@
1847     \global\instanzaLfalse
1848     \global\instanzaRfalse
1849     \ignorespaces}
1850

```

\ledstrutL Struts inserted into leftand right text lines.

```

\ledstrutR 1851 \newcommand*{\ledstrutL}{\strut}
1852 \newcommand*{\ledstrutR}{\strut}
1853

```

\cleartoevenpage \cleartoevenpage, which is defined in the memoir class, is like \clear(double)page except that we end up on an even page. \cleartol@devenpage is similar except that it first checks to see if it is already on an empty page. \clearl@dleftpage \clearl@drighthpage

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

```

1854 \providecommand{\cleartoevenpage}[1][\@empty]{%
1855   \clearpage
1856   \ifodd\c@page\hbox{#1\clearpage\fi}
1857 \newcommand*{\cleartol@devenpage}{%
1858   \ifdim\pagetotal<\topskip% on an empty page
1859   \else
1860     \clearpage
1861   \fi
1862   \ifodd\c@page\hbox{}\clearpage\fi}
1863 \newcommand*{\clearl@dleftpage}{%
1864   \clearpage
1865   \ifodd\c@page\else
1866     \led@err@LeftOnRightPage
1867     \hbox{}%
1868     \cleardoublepage
1869   \fi}
1870 \newcommand*{\clearl@drighthouse}{%
1871   \clearpage
1872   \ifodd\c@page
1873     \led@err@RightOnLeftPage
1874     \hbox{}%
1875     \cleartoevenpage
1876   \fi}
1877

```

`\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`. Similarly for `\getlinesfromparlistR`.

```

\cs@linesinparR 1878 \newcommand*{\getlinesfromparlistL}{%
1879   \ifx\linesinpar@listL\empty
1880     \gdef\cs@linesinparL{0}%
1881   \else
1882     \gl@p\linesinpar@listL\to\cs@linesinparL
1883   \fi}
1884 \newcommand*{\getlinesfromparlistR}{%
1885   \ifx\linesinpar@listR\empty
1886     \gdef\cs@linesinparR{0}%
1887   \else
1888     \gl@p\linesinpar@listR\to\cs@linesinparR
1889   \fi}
1890

```

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

```

\cs@linesonpageR 1891 \newcommand*{\getlinesfrompagelistL}{%
1892   \ifx\linesonpage@listL\empty

```

```

1893 \gdef\@cs@linesonpageL{1000}%
1894 \else
1895 \gl@p\linesonpage@listL\to\@cs@linesonpageL
1896 \fi}
1897 \newcommand*\getlinesfrompagelistR{%
1898 \ifx\linesonpage@listR\empty
1899 \gdef\@cs@linesonpageR{1000}%
1900 \else
1901 \gl@p\linesonpage@listR\to\@cs@linesonpageR
1902 \fi}
1903

```

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

```

1904 \newcommand*\@writelinesonpageL[1]{%
1905 \edef\next{\write\linenum@out{\string\@lopL{#1}}}%
1906 \next}
1907 \newcommand*\@writelinesonpageR[1]{%
1908 \edef\next{\write\linenum@outR{\string\@lopR{#1}}}%
1909 \next}
1910

```

\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>.

```

1911 \newcommand*\l@dcalc@maxoftwo[3]{%
1912 \ifnum #2>#1\relax
1913 #3=#2\relax
1914 \else
1915 #3=#1\relax
1916 \fi}
1917 \newcommand*\l@dcalc@minoftwo[3]{%
1918 \ifnum #2<#1\relax
1919 #3=#2\relax
1920 \else
1921 #3=#1\relax
1922 \fi}
1923

```

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

```

\checkpageL 1924 \newif\ifl@dsamepage
\checkpageR 1925 \l@dsamepagetrue

```

```

1926 \newif\ifl@dpagfull
1927 \newcommand*\checkpageL{%
1928   \l@dpagfulltrue
1929   \l@dsamepagetrue
1930   \check@goal
1931   \ifdim\pagetotal<\ledthegoal
1932     \ifnum\numpagelinesL<\l@dminpagelines
1933       \else
1934         \l@dsamepagefalse
1935         \l@dpagfullfalse
1936       \fi
1937     \else
1938       \l@dsamepagefalse
1939       \l@dpagfulltrue
1940     \fi}
1941 \newcommand*\checkpageR{%
1942   \l@dpagfulltrue
1943   \l@dsamepagetrue
1944   \check@goal
1945   \ifdim\pagetotal<\ledthegoal
1946     \ifnum\numpagelinesR<\l@dminpagelines
1947       \else
1948         \l@dsamepagefalse
1949         \l@dpagfullfalse
1950       \fi
1951     \else
1952       \l@dsamepagefalse
1953       \l@dpagfulltrue
1954     \fi}
1955

```

`\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`.

```

1956 \newdimen\ledthegoal
1957 \ifshiftedpstarts
1958   \newcommand*\goalfraction{0.95}
1959 \else
1960   \newcommand*\goalfraction{0.9}
1961 \fi
1962
1963 \newcommand*\check@goal{%
1964   \ledthegoal=\goalfraction\pagegoal}
1965

```

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

```

\ifwrittenlinesL 1966 \newif\ifwrittenlinesL
                  1967 \newif\ifwrittenlinesR
                  1968

```

`\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.

```

1969 \newcommand*{\get@nextboxL}{%
1970   \ifvbox\namebox{1@dLcolrawbox\the\1@dpscL}% box is not empty

   The current box is not empty; do nothing.

1971   \else%                                     box is empty

   The box is empty; check if enough lines (real and blank) have been output.
1972   \ifnum\usenamecount{1@dmaxlinesinpar\the\1@dpscL}>\@donetotallinesL
1973   \else

   Sufficient lines have been output.

1974   \ifwrittenlinesL
1975   \else

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

1976   \@writelinesinparL
1977   \writtenlinesLtrue
1978   \fi
1979   \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.

1980   \writtenlinesLfalse
1981   \ifbypstart@
1982   \ifnum\value{pstartL}<\value{pstartLold}
1983   \else
1984     \global\line@num=0
1985     \resetprevline@
1986   \fi
1987   \fi
1988   \addtocounter{pstartL}{1}
1989   \global\pstartnumtrue
1990   \1@dcalc@maxoftwo{\the\usenamecount{1@dmaxlinesinpar\the\1@dpscL}}%
1991     {\the\@donetotallinesL}%
1992     {\usenamecount{1@dmaxlinesinpar\the\1@dpscL}}%
1993   \global\@donetotallinesL \z@
1994   \global\advance\1@dpscL \@ne
1995   \fi
1996   \fi
1997 \fi}

1998 \newcommand*{\get@nextboxR}{%
1999   \ifvbox\namebox{1@dRcolrawbox\the\1@dpscR}% box is not empty
2000   \else%                                     box is empty
2001   \ifnum\usenamecount{1@dmaxlinesinpar\the\1@dpscR}>\@donetotallinesR
2002   \else
2003   \ifwrittenlinesR
2004   \else

```

```

2005         \@writelinesinparR
2006         \writtenlinesRtrue
2007     \fi
2008     \ifnum\l@dnumstartsR>\l@dpscR
2009         \writtenlinesRfalse
2010         \ifbypstartR
2011             \ifnum\value{pstartR}<\value{pstartRold}
2012             \else
2013                 \global\line@numR=0
2014                 \resetprevline@
2015             \fi
2016         \fi
2017         \addtocounter{pstartR}{1}
2018         \global\pstartnumRtrue
2019         \l@dcalc@maxoftwo{\the\usernamecount{l@dmaxlinesinpar\the\l@dpscR}}%
2020                             {\the\@donetotallinesR}%
2021                             {\usernamecount{l@dmaxlinesinpar\the\l@dpscR}}%
2022         \global\@donetotallinesR \z@
2023         \global\advance\l@dpscR \@ne
2024     \fi
2025 \fi
2026 \fi}
2027

```

25 The End

i/codei

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>	1481, 1482, 1486, 1503, 1517
<code>\@M</code>	1492
<code>\@adv</code>	<u>352</u> , 621, 622
<code>\@afterindentfalse</code>	730
<code>\@arabic</code>	209, 210, 779, 782
<code>\@astanza@line</code>	1502, 1508, <u>1511</u>
<code>\@auxout</code>	1302, 1314, 1591
<code>\@chapter</code>	731
<code>\@cs@linesinparL</code>	1767, <u>1878</u>
<code>\@cs@linesinparR</code>	1767, <u>1878</u>
<code>\@cs@linesonpageL</code>	1775, 1835, <u>1891</u>
<code>\@cs@linesonpageR</code>	1775, 1835, <u>1891</u>
<code>\@currentlabel</code>	814, 847
<code>\@donereallinesL</code>	
.	<u>891</u> , 920, 1729, 1731, 1781
<code>\@donereallinesR</code>	
.	<u>891</u> , 955, 1734, 1736, 1783
<code>\@donetotallinesL</code>	<u>891</u> ,
.	921, 924, 1782, 1972, 1991, 1993
<code>\@donetotallinesR</code>	<u>891</u> ,
.	956, 959, 1784, 2001, 2020, 2022
<code>\@insertR</code>	1256–1258, 1271–1273
<code>\@l</code>	<u>275</u> , 590
<code>\@l@dttempcnta</code>	425,
	427, 429, 430, 434, 436, 438,
	439, 1009, 1048, 1049, 1051,
	1053, 1056, 1057, 1074–1078,
	1080, 1087, 1092, 1096, 1104,
	1109, 1113, 1146, 1149, 1151, 1155
<code>\@l@dttempcntb</code>	168, 170, 172, 1039,
	1040, 1087, 1092, 1096, 1104,
	1109, 1113, 1138, 1142, 1155,
	1163–1165, 1167, 1187–1189,
	1191, 1208–1210, 1212, 1343,
	1345, 1347, 1411–1413, 1415,
	1545–1549, 1553–1556, 1560–1563
<code>\@l@reg</code>	324
<code>\@l@regR</code>	<u>275</u>
<code>\@lab</code>	544, 1293, 1305, <u>1329</u>
<code>\@lock</code>	907, 990
<code>\@lockR</code>	60, 297, 299, 301, 314,
	459, 475, 476, 478, 479, 507,
	508, 510, 942, 973, 1015, 1017,
	1018, 1020, 1101, 1118, 1120, 1122
<code>\@lopL</code>	<u>568</u> , 1905
<code>\@lopR</code>	<u>568</u> , 1908
<code>\@nameuse</code>	1439, 1443

`\@nobreakfalse` 788, 821
`\@nobreaktrue` 786, 790, 819, 823
`\@oldnobreak` 786, 788, 819, 821, 861, 877
`\@pend` 559, 1729
`\@pendR` 559, 1734
`\@pstartfalse` 1702
`\@pstarttrue` 1702
`\@ref` 531, 594, 598
`\@ref@reg` 557
`\@schapter` 731
`\@set` 384, 628, 629, 1676, 1680
`\@tag` 658, 674
`\@temp` 1582
`\@templ@ed` 1402, 1403
`\@writelinesinparL` .. 1672, 1727, 1976
`\@writelinesinparR` .. 1673, 1727, 2005
`\@writelinesonpageL` . 1805, 1807, 1904
`\@writelinesonpageR` . 1825, 1827, 1904
`\@xloop` 1269

A

`\absline@num` 418, 432, 451, 981
`\absline@numR` 58, 226, 277, 280, 283,
 415, 423, 444, 463, 497, 525,
 536, 964, 1001, 1002, 1039, 1255
`\actionlines@list`
 267, 270, 418, 432, 451
`\actionlines@listR`
 230, 245, 259, 262, 415,
 423, 444, 463, 497, 525, 1061, 1064
`\actions@list` . 271, 419, 439, 453, 455
`\actions@listR`
 . 230, 246, 263, 416, 430, 446,
 448, 465, 474, 499, 506, 526, 1065
`\add@inserts` 913
`\add@inserts@nextR` 1244
`\add@insertsR` 948, 1244
`\add@penaltiesL` 919, 1265
`\add@penaltiesR` 954, 1265
`\addtocontents` 1592–1594
`\addtocounter`
 . 863, 879, 1683, 1684, 1988, 2017
`\advancelabel@refs` 1300, 1312
`\advanceline` 620, 651
`\affixline@num` 911
`\affixline@numR` 946, 1071
`\affixpstart@numL` 910, 1177
`\affixpstart@numR` 945, 1177
`\affixside@note` 914
`\affixside@noteR` 949, 1389

`\appto` 1395, 1396
`\apptocmd` 1622
`\araw@textfalse` 1714
`\araw@texttrue` 1714
`astanza (environment)` 9, 1484
`\AtBeginDocument` ... 1325, 1567, 1602

B

`\ballast@count` 999, 1004
`\bbl@main@language` 1618, 1619
`\bbl@set@language` 1609, 1610
`\beginnumbering` .. 7, 36, 737, 755, 793
`\beginnumberingR` ... 49, 117, 755, 826
`\bfseries` 779, 782
`\bypage@Rfalse` 137, 152, 157
`\bypage@Rtrue` 137, 147
`\bypstart@Rfalse` 137, 148, 158
`\bypstart@Rtrue` 137, 153

C

`\c@ballast` 1004
`\c@chapter` 100
`\c@chapterR` 100
`\c@firstlinenumR` 177, 1144
`\c@firstsublinenumR` 181, 1139
`\c@linenumincrementR` 177, 1144
`\c@page` ... 590, 1856, 1862, 1865, 1872
`\c@pstartL` 779
`\c@pstartR` 782
`\c@section` 101
`\c@sectionR` 101
`\c@sublinenumincrementR` .. 181, 1139
`\c@subsection` 102
`\c@subsectionR` 102
`\c@subsubsection` 103
`\c@subsubsectionR` 103
`\ch@ck@l@ckR` 1071
`\ch@cksub@l@ckR` 1071
`\ch@cksub@lockR` 1140
`\chapter` 717, 718, 726
`\chapterinpages` 710, 718, 728
`\chardef` 1481
`\check@goal` 1930, 1944, 1956
`\check@pstarts`
 1647, 1674, 1702, 1761, 1769, 1777
`\checkpageL` 1788, 1802, 1924
`\checkpageR` 1811, 1822, 1924
`\checkraw@text`
 1653, 1670, 1714, 1785, 1831
`\cleardoublepage` 1868

- `\clearl@leftpage` 1810, 1854
`\clearl@drighthpage` 1830, 1854
`\cleartoevenpage` 1854
`\cleartol@evenpage` 1750, 1854
`\closeout` 581, 585
`\columnrulewidth` 5, 1697
`\Columns` 5, 1633
`\columnseparator` 5, 1667, 1697
`\correcthangingL` 916, 1462
`\correcthangingR` 951, 1462
`\countLline` 886, 897
`\countRline` 886, 932
`\critext` 656
- ### D
- `\DeclareOption` 8, 9
`\def@tempb` 155
`\dimen` 607, 608, 612–614, 618
`\divide` 1076
`\do@actions` 982
`\do@actions@fixedcodeR` 1008
`\do@actions@nextR` 1008
`\do@actionsR` 965, 1008
`\do@ballast` 983
`\do@ballastR` 966, 999
`\do@lineL` 896, 1657, 1661, 1793
`\do@lineLhook` 901, 928
`\do@lineR` 931, 1658, 1663, 1814
`\do@lineRhook` 928, 936
`\do@lockoff` 494
`\do@lockoffL` 518
`\do@lockoffR` 494
`\do@lockon` 459
`\do@lockonL` 491
`\do@lockonR` 459
`\dolistloop` 1400
`\dummy@ref` 540
- ### E
- `\edfont@info` 693, 696, 702, 705
`\edlabel` 1291
`\edtext` 672
`\eledmac@error` 21, 24, 27, 29
`\eledmac@warning` 1401
`\empty` .. 81, 84, 259, 267, 666, 682,
 691, 700, 802, 835, 1061, 1143,
 1151, 1246–1248, 1259, 1270,
 1294, 1306, 1879, 1885, 1892, 1898
`\end@lemmas` 666, 667, 682, 683
`\endashchar` 1281
- `\endgraf` 857, 873, 1641, 1753
`\endline@num` 547, 553
`\endlock` 640, 1490, 1499, 1504
`\endnumbering` 7, 39, 74, 121, 756
`\endnumberingR` 52, 74, 106, 116, 129, 756
`\endpage@num` 546, 553
`\endstanzaextra` 1506
`\endsub` 607
`\endsubline@num` 548, 554
environments:
 astanza 9, 1484
 Leftside 6, 735
 pages 5, 710
 pairs 5, 710
 Rightside 6, 753
`\extensionchars` . 47, 66, 112, 126, 134
- ### F
- `\f@x@l@cksR` 1071
`\first@linenum@out@Rfalse` .. 576, 582
`\first@linenum@out@Rtrue` 576
`\firstlinenum` 6, 186
`\firstsublinenum` 6, 186
`\fix@page` 320, 327
`\flag@end` 591, 671, 687
`\flag@start` 591, 663, 679
`\flush@notes` 1686, 1840
`\flush@notesR` 1268, 1687, 1841
`\fullstop` . 222, 1278, 1280, 1282, 1284
- ### G
- `\get@linelistfile` 255
`\get@nextboxL` 1801, 1969
`\get@nextboxR` 1821, 1969
`\getline@numL` 906, 980
`\getline@numR` 941, 963
`\getlinesfrompagelistL`
 1773, 1833, 1891
`\getlinesfrompagelistR`
 1774, 1834, 1891
`\getlinesfromparlistL` ... 1765, 1878
`\getlinesfromparlistR` ... 1766, 1878
`\gl@p` 262, 263,
 270, 271, 667, 683, 695, 704,
 1064, 1065, 1252, 1256, 1271,
 1297, 1309, 1882, 1888, 1895, 1901
`\goalfraction` 6, 1956
- ### H
- `\hangingsymbol` 10, 1453, 1459

\hb@xt@ ... 909, 916, 923, 944, 951,
958, 1665, 1796, 1798, 1817, 1819
\hsize 812,
845, 1665, 1796, 1798, 1817, 1819

I

\if@filesw 1590
\if@firstcolumn 1157, 1181, 1202, 1405
\if@nobreak 785, 818
\if@pstarts 1648, 1702, 1762, 1778
\ifaraw@text ... 1655, 1714, 1787, 1832
\ifautopar 811, 844
\ifbypage@ 343
\ifbypage@R 137, 333, 1043
\ifbypstart@ 561, 1675, 1981
\ifbypstart@R ... 137, 565, 1679, 2010
\ifdim 608, 612, 614,
618, 1796, 1817, 1858, 1931, 1945
\iffirst@linenum@out@R 576, 580
\ifinserthangingsymbol .. 1451, 1465
\ifinserthangingsymbolR
..... 1449, 1457, 1475
\ifinstanzaL 733, 733, 1452, 1464
\ifinstanzaR 733, 734, 1458, 1474
\ifl@d@dash 1281
\ifl@d@elin 1283, 1284
\ifl@d@esl 1284
\ifl@d@pnun 1278, 1282
\ifl@d@ssub 1280
\ifl@dpagefull 1804, 1824, 1924
\ifl@dpadding 11, 1463, 1473
\ifl@dpairing 11, 78
\ifl@dsamelang 1578, 1656
\ifl@dsamepage 1791, 1813, 1924
\ifl@dskipnumber 1134
\ifl@dusedbabel 1576
\iflabelpstart 814, 847
\ifledplinenun 1279
\ifledRcol 11, 169, 191,
195, 199, 203, 242, 257, 321,
330, 354, 368, 385, 402, 414,
422, 443, 488, 515, 524, 533,
592, 602, 609, 615, 621, 628,
636, 641, 645, 650, 660, 676,
690, 1292, 1330, 1344, 1355,
1366, 1378, 1425, 1438, 1613, 1623
\ifnoteschanged@ 88
\ifnumberedpar@ ... 795, 828, 853,
869, 1354, 1365, 1377, 1424, 1437
\ifnumbering 37, 142, 791, 850

\ifnumberingR ... 50, 75, 108, 824, 866
\ifnumberline 967, 984, 1133
\ifnumberpstart ... 811, 844, 862, 878
\ifnumequal 1393
\ifnumgreater 1401
\ifodd 1167, 1191,
1212, 1415, 1856, 1862, 1865, 1872
\ifpst@rtedL 32, 799
\ifpst@rtedR 32, 832
\ifpstartnum 1222, 1227
\ifpstartnumR 1177
\ifshiftedpstarts 5, 1795, 1816, 1957
\ifsidepstartnum 811, 844, 1179, 1200
\ifsublines@ 220,
309, 353, 386, 393, 424, 433,
445, 452, 464, 498, 552, 554,
968, 985, 1050, 1137, 1332, 1336
\ifvbox 898, 933, 1718, 1721, 1970, 1999
\ifvmode 1299, 1311
\ifwrittenlinesL 1966, 1974
\ifwrittenlinesR 1967, 2003
\initnumbering@reg 45
\initnumbering@sectcmd 70
\initnumbering@sectcountR 71, 95
\insert@count 530, 598, 661,
677, 1361, 1373, 1385, 1432, 1445
\insert@countR 531, 594,
660, 676, 1369, 1381, 1428, 1441
\inserthangingsymbolfalse 907
\inserthangingsymbolL 916, 1449
\inserthangingsymbolR 951, 1449
\inserthangingsymbolRfalse 942
\inserthangingsymbolRtrue 942
\inserthangingsymboltrue 907
\insertlines@listR
.... 81, 230, 244, 536, 1248, 1252
\inserts@list
801, 1360, 1372, 1384, 1431, 1444
\inserts@listR
.. 834, 1243, 1246, 1256, 1270,
1271, 1357, 1368, 1380, 1427, 1440
\instanzaLfalse 1694, 1847
\instanzaLtrue 744
\instanzaRfalse 1695, 1848
\instanzaRtrue 766
\interlinepenalty 1492
\itemcount@ 1391, 1393, 1398, 1401

L

\l@d@nums 693, 696, 702, 705

- \l@d@set 401, 636, 637
- \l@dbbl@set@language 1587, 1610
- \l@dbfnote 1423
- \l@dc@maxchunks 807, 809,
840, 842, 1535, 1545, 1553, 1560
- \l@dcalc@maxoftwo
..... 1767, 1911, 1990, 2019
- \l@dcalc@minoftwo .. 1775, 1835, 1911
- \l@dcalcnun 1071
- \l@dchecklang 1580, 1654
- \l@dchset@num 276, 279, 401
- \l@dcsnote 1353
- \l@dcsnotetext 1400, 1403
- \l@demptyd@ta 902, 937
- \l@dend@stuff ... 48, 67, 113, 127, 135
- \l@dgetline@margin 167
- \l@dgetsidenote@margin 1342
- \l@dld@ta 912, 947,
1158, 1170, 1182, 1194, 1203, 1215
- \l@dleftbox
.. 883, 908, 923, 1666, 1796, 1798
- \l@dlinenumR 212
- \l@dlsn@te 915, 950
- \l@dlsnote 1353
- \l@dmake@labels 1315
- \l@dmake@labelsR 1303, 1319
- \l@dminpagelines
.... 1738, 1776, 1836, 1932, 1946
- \l@dnumstartsl . 41, 806, 807, 809,
811, 1539, 1571, 1636, 1637,
1691, 1705, 1747, 1748, 1845, 1979
- \l@dnumstartsr . 54, 839, 840, 842,
844, 1539, 1572, 1636, 1637,
1692, 1708, 1747, 1748, 1846, 2008
- \l@doldbbl@set@language 1609
- \l@doldselectlanguage 1608, 1612, 1617
- \l@dpagetrue 1924
- \l@dpagetrue 1924
- \l@dpagingfalse 13, 712, 725
- \l@dpagingtrue 720
- \l@dpairingfalse 11, 714, 724
- \l@dpairingtrue 711, 719
- \l@dpscL 898, 903, 1541, 1573, 1645,
1651, 1689, 1705, 1718, 1757,
1763, 1768, 1771, 1779, 1843,
1970, 1972, 1979, 1990, 1992, 1994
- \l@dpscR 933, 938, 1542, 1574,
1646, 1652, 1690, 1708, 1721,
1758, 1764, 1772, 1780, 1844,
1999, 2001, 2008, 2019, 2021, 2023
- \l@drd@ta 916, 951,
1160, 1168, 1184, 1192, 1205, 1213
- \l@dtrightbox
.. 883, 943, 958, 1668, 1817, 1819
- \l@drsn@te 917, 952
- \l@drsnote 1353
- \l@dsamelangfalse 1578, 1581
- \l@dsamelangtrue 1578, 1584
- \l@dsamepagefalse 1924
- \l@dsamepagetrue 1924
- \l@dsetupmaxlinecounts .. 1552, 1569
- \l@dsetupprwboxes 1544, 1568
- \l@dskipnumberfalse 1135
- \l@dskipnumbertrue 1031
- \l@dunhbox@line 916, 951
- \l@dusedbabelfalse 1576, 1605
- \l@dusedbabeltrue 1576, 1607
- \l@duselanguage
.... 1597, 1660, 1662, 1789, 1812
- \l@dzeromaxlinecounts ... 1552, 1570
- \l@dzeropenalties 856, 872, 1640, 1752
- \l@pscL 1541
- \l@pscR 1541
- \label@refs
1295, 1297, 1303, 1307, 1309, 1315
- \labelref@list 1306, 1309, 1337
- \labelref@listR 1289, 1294, 1297, 1333
- \language name .. 1588, 1589, 1591–1594
- \last@page@num 341, 347
- \last@page@numR 327
- \lastbox 905, 940
- \lastskip 607, 613
- \lcolwidth 5, 6, 15, 721, 812, 909, 923
- \led@err@BadLeftRightPstarts ...
..... 23, 1637, 1748
- \led@err@LeftOnRightPage ... 26, 1866
- \led@err@LineationInNumbered ... 143
- \led@err@NumberingNotStarted ... 92
- \led@err@NumberingShouldHaveStarted
..... 115
- \led@err@NumberingStarted 38, 51
- \led@err@PendNoPstart 854, 870
- \led@err@PendNotNumbered ... 851, 867
- \led@err@PstartInPstart ... 796, 829
- \led@err@PstartNotNumbered . 792, 825
- \led@err@RightOnLeftPage ... 26, 1873
- \led@err@TooManyPstarts 20, 808, 841
- \led@mess@NotesChanged 89
- \led@mess@SectionContinued
..... 111, 125, 133

- \led@warn@BadAction 1033
 - \led@warn@BadAdvancelineLine 371, 377
 - \led@warn@BadAdvancelineSubline .
..... 357, 363
 - \led@warn@BadLineation 160
 - \led@warn@BadSetline 626
 - \led@warn@BadSetlinenum 634
 - \led@warn@DuplicateLabel 1321
 - \ledllfill 916, 951
 - \ledRcolfalse 14, 736, 768
 - \ledRcoltrue 754
 - \ledrlfill 916, 951
 - \ledsavedprintlines 8, 1276
 - \ledstrutL 1796, 1798, 1851
 - \ledstrutR 1817, 1819, 1851
 - \ledthegoal 1931, 1945, 1956
 - \leftlinenumR 212, 1158, 1170
 - \leftpstartnumL 1177
 - \leftpstartnumR 1177
 - Leftside (environment) 6, 735
 - \Leftsidehook 742, 748
 - \Leftsidehookend 747, 748
 - \line@list 700, 704
 - \line@list@stuff 47, 126
 - \line@list@stuffR .. 66, 112, 134, 578
 - \line@listR . 84, 230, 243, 554, 691, 695
 - \line@margin 172, 1187
 - \line@marginR 165, 1163, 1208
 - \line@num . 344, 375, 376, 378, 396,
407, 408, 436, 561, 991, 1335, 1984
 - \line@numR 59, 219,
226, 281, 315, 334, 369, 370,
372, 389, 403, 404, 427, 547,
551, 565, 974, 1044, 1053, 1142,
1144, 1146, 1147, 1331, 1401, 2013
 - \lineation 763
 - \lineationR 141, 763
 - \linenum@out 597,
605, 610, 616, 622, 629, 637,
642, 646, 1305, 1676, 1729, 1905
 - \linenum@outR 575, 581,
583, 585, 586, 590, 593, 603,
609, 615, 621, 628, 636, 641,
645, 650, 1293, 1680, 1734, 1908
 - \linenumberlist 1143, 1147
 - \linenumincrement 6, 186
 - \linenummargin 165
 - \linenumr@p 1279, 1283, 1331, 1335
 - \linenumrepR 209, 219
 - \linenumsep
. 214, 216, 1224, 1227, 1236, 1239
 - \linesinpar@listL
..... 235, 251, 562, 1879, 1882
 - \linesinpar@listR
..... 235, 247, 566, 1885, 1888
 - \linesonpage@listL 252, 570, 1892, 1895
 - \linesonpage@listR 248, 573, 1898, 1901
 - \list@clear
. 243–248, 251, 252, 254, 801, 834
 - \list@clearing@reg 250
 - \list@create
... 230–233, 235–237, 1243, 1289
 - \lock@disp 1103, 1107, 1112
 - \lock@off 485, 486, 494, 645, 646
 - \lock@on 641, 642
- M**
- \maxchunks 4, 1535
 - \maxlinesinpar@list 235, 254
 - \memorydump 8, 741, 759
 - \memorydumpL 120, 741
 - \memorydumpR 120, 759
 - \message 46, 65
 - \multiply 1077
- N**
- \n@num 522, 650
 - \n@num@reg 528
 - \namebox 898, 903, 933,
938, 1519, 1718, 1721, 1970, 1999
 - \NeedsTeXFormat 2
 - \newline 916
 - \newlineR 589, 951
 - \newbox 774, 883, 884, 1520
 - \newcounter 96–99, 177,
179, 181, 183, 777, 778, 780, 781
 - \newif . 5, 12, 33, 137, 138, 576, 733,
734, 1231, 1449, 1576, 1578,
1702, 1714, 1924, 1926, 1966, 1967
 - \newnamebox 1519, 1547, 1548
 - \newnamecount 1530, 1555
 - \newwrite 575
 - \next@action 271
 - \next@actionline 268, 270
 - \next@actionlineR
. 260, 262, 1002, 1040, 1062, 1064
 - \next@actionR 263, 1003,
1041, 1042, 1047, 1048, 1056, 1065
 - \next@insert 802

- \next@insertR 835, 1247, 1250, 1252, 1255, 1259
 - \next@page@num 348, 419
 - \next@page@numR 63, 284, 286, 338, 416
 - \no@expands 658, 674
 - \normal@pars 77, 805, 838
 - \normalbfnoteX 1436
 - \noteschanged@true 82, 85, 692, 701, 1249
 - \num@lines 857, 1641, 1753
 - \num@linesR 773, 873, 1642, 1754
 - \numberedpar@true 813, 846
 - \numberingRfalse 76
 - \numberingRtrue 56, 106, 130
 - \numberingtrue 43, 122
 - \numberpstartfalse 9
 - \numberpstarttrue 9
 - \numdef 1391, 1398
 - \numlabfont 219
 - \numpagelinesL 1738, 1794, 1805, 1809, 1932
 - \numpagelinesR 1738, 1815, 1825, 1829, 1946
- O**
- \oldchapter 717, 726
 - \oldstanza 743, 744, 746, 765, 766, 769
 - \one@line 903, 905, 916
 - \one@lineR 773, 938, 940, 951
 - \openout 583, 586
- P**
- \p@pstartL 815
 - \p@pstartR 848
 - \page@action 285, 413, 541
 - \page@num 266, 346, 1189, 1413
 - \page@numR 239, 258, 336, 546, 551, 1042, 1165, 1210, 1401
 - \pagegoal 1964
 - \Pages 5, 1742
 - pages (environment) 5, 710
 - \pagetotal 1858, 1931, 1945
 - pairs (environment) 5, 710
 - \par@line 858, 1643, 1755
 - \par@lineR 773, 874, 1644, 1756
 - \pausenumbering 757
 - \pausenumberingR 105, 757
 - \pend 6, 740, 762, 797, 1505
 - \pendL 740, 850
 - \pendR 762, 830, 866
- \prevgraf 857, 873, 1641, 1642, 1753, 1754
 - \printlines 1287
 - \printlinesR 8, 1276
 - \ProcessOptions 10
 - \protected@edef 814, 847
 - \protected@write ... 1302, 1314, 1591
 - \ProvidesPackage 3
 - \pst@rtedLfalse 32, 42
 - \pst@rtedLtrue 123, 803
 - \pst@rtedRfalse 34, 55, 79
 - \pst@rtedRtrue 109, 131, 836
 - \pstart 6, 21, 25, 738, 761, 1507
 - \pstartL 738, 776
 - \pstartnumfalse 1224, 1229
 - \pstartnumRfalse 1236, 1241
 - \pstartnumRtrue 1232, 1650, 2018
 - \pstartnumtrue 1649, 1989
 - \pstartR 761, 776
- R**
- \Rcolwidth 5, 6, 15, 722, 845, 944, 958
 - \read@linelist 241, 579
 - \rem@inder 1147, 1149–1151
 - \resetprevline@ 1677, 1681, 1985, 2014
 - \resumenumbering 758
 - \resumenumberingR 105, 758
 - \rightlinenumR 212, 1160, 1168
 - \rightpstartnumL 1177
 - \rightpstartnumR 1177
 - Rightside (environment) 6, 753
 - \Rightsidehook 748, 764
 - \Rightsidehookend 748, 770
 - \rlap 1160, 1168, 1184, 1192, 1205, 1213
 - \Rlineflag 8, 207, 219, 1279, 1283, 1323
 - \rule 1698
- S**
- \sc@n@list 1148, 1150
 - \secdef 731
 - \section@num 44, 46, 47, 124–126
 - \section@numR 30, 57, 65, 66, 110–112, 132–134
 - \select@language ... 1589, 1591–1594
 - \selectlanguage 1597
 - \set@line 659, 675, 689
 - \set@line@action 278, 382, 391, 398, 421, 543
 - \setl@dlp@rbox 1406, 1418
 - \setl@drp@rbox 1408, 1416

- \setline 624
 - \setlinenum 632
 - \setnamebox 811, 844, 1519
 - \setprintlines 1277
 - \shiftedpstartsfalse 7
 - \shiftedpstartstrue 6, 8, 9
 - \shiftedversesfalse 7
 - \shiftedversestrue 6
 - \showlemma 665, 681
 - \sidenote@margin 1347, 1351
 - \sidenote@marginR 1340, 1411
 - \sidenotecontent@
 - 1390, 1395, 1396, 1406, 1408, 1418
 - \sidenotecontent@t 1416
 - \sidenotemargin 1340
 - \sidenotesep 1396
 - \skip@lockoff 486, 494
 - \skipnumbering 9, 649
 - \skipnumbering@reg 653
 - \smash 1698
 - \splittopskip 900, 935
 - \stanza ... 743, 744, 746, 765, 766, 769
 - \stanza@count 1487, 1501, 1512
 - \stanza@chang 1489, 1514
 - \stanzaindentbase .. 1467, 1477, 1512
 - \startlock 640
 - \startstanzahook 1485
 - \startsub 607
 - \sub@action 294, 442, 542
 - \sub@change 64, 288, 289, 295
 - \sub@lock 986
 - \sub@lockR 61, 303, 305, 307,
 - 310, 460, 466, 467, 469, 470,
 - 500, 501, 503, 969, 1023, 1025,
 - 1026, 1028, 1084, 1124, 1126, 1128
 - \sub@off 615, 616
 - \sub@on 609, 610
 - \subline@num 221, 344, 361,
 - 362, 364, 394, 434, 987, 992, 1336
 - \subline@numR 222,
 - 226, 311, 315, 334, 355, 356,
 - 358, 387, 425, 548, 552, 970,
 - 975, 1044, 1051, 1138, 1139, 1332
 - \sublinenumincrement 6, 186
 - \sublinenumr@p . 1280, 1284, 1332, 1336
 - \sublinenumrepR 209, 222
 - \sublines@false 62, 292, 1013
 - \sublines@true 290, 1011
 - \sublock@disp 1086, 1090, 1095
 - \symplinenum 1279
 - \sza@penalty 1496, 1500
- T**
- \textwidth 16, 18, 721, 722
 - \theledlanguageL 1582, 1597, 1660, 1789
 - \theledlanguageR 1582, 1597, 1662, 1812
 - \thepage 590, 1303, 1315
 - \thepstart 739, 760
 - \thepstartL
 - 9, 739, 779, 811, 815, 1223, 1228
 - \thepstartR
 - 9, 760, 782, 844, 848, 1235, 1240
 - \thr@@ .. 469, 478, 501, 508, 1018, 1026
 - \topskip 1858
- U**
- \unhbox 1524,
 - 1666, 1668, 1796, 1798, 1817, 1819
 - \unhnamebox 1519
 - \unvbox 905, 940, 1526
 - \unvnamebox 1519
 - \usernamecount
 - 1488, 1495, 1530, 1562, 1768,
 - 1972, 1990, 1992, 2001, 2019, 2021
- V**
- \value 800, 833,
 - 1634, 1635, 1743, 1744, 1982, 2011
 - \vbadness 899, 934
 - \vbfnoteX 1439, 1443
 - \vbox 811, 844
 - \vl@dbfnote 1426, 1430
 - \vl@dcsnote 1379, 1383
 - \vl@dlsnote 1356, 1359
 - \vl@drsnote 1367, 1371
 - \vsplit 903, 938
- W**
- \wd 916, 951
 - \writtenlinesLfalse 1759, 1980
 - \writtenlinesLtrue 1977
 - \writtenlinesRfalse 1760, 2009
 - \writtenlinesRtrue 2006
- X**
- \x@lemma 667–669, 683–685
 - \xpg@main@language 1627, 1628
 - \xpg@set@language 1622, 1626
 - \xright@appenditem
 - 415, 416, 418, 419, 423,

430, 432, 439, 444, 446, 448,
451, 453, 455, 463, 465, 474,
497, 499, 506, 525, 526, 536,
550, 562, 566, 570, 573, 1331,

1335, 1356, 1359, 1367, 1371,
1379, 1383, 1426, 1430, 1439, 1443

Z

`\zz@@@` 1295, 1307

Change History

v0.1	General: First public release 1	hangingsymbol insertion, preventing undesirable insertions. 53
v0.10	General: <code>\edlabel</code> commands on the right side are now correctly indicated. 1	v0.2
	<code>\edlabel</code> commands which start a paragraph are now put in the right place. 1	General: Added section of babel related code 57
v0.11	General: Change <code>\do@lineL</code> and <code>\do@lineR</code> to allow line numbering by <code>pstart</code> (like in <code>eledmac 0.15</code>). 38	Fix babel problems 1
	Lineation can be by <code>pstart</code> (like in <code>eledmac 0.15</code>). 16	<code>\Columns</code> : Added <code>\l@dcchecklang</code> and <code>\l@duselanguage</code> to <code>\Columns</code> 60
	New management of hangingsymbol insertion, preventing undesirable insertions. 53	<code>\Pages</code> : Added <code>\l@duselanguage</code> to <code>\Pages</code> 64
	Prevent shift of column separator when a verse is hanged 53	v0.3
	<code>\affixline@numR</code> : Changed <code>\affixline@numR</code> to allow to disable line numbering (like in <code>eledmac 0.15</code>). 43	General: Reorganize for <code>ledarab</code> .. 1
	<code>\Columns</code> : Line numbering by <code>pstart</code> 60	<code>\affixline@numR</code> : Changed <code>\affixline@numR</code> to match new <code>eledmac</code> 43
	<code>\get@nextboxR</code> : Change <code>\get@nextboxL</code> and <code>\get@nextboxR</code> to allow to disable line numbering (like in <code>eledmac 0.15</code>). 69	<code>\do@actions@nextR</code> : Used <code>\do@actions@fixedcode</code> in <code>\do@actionsR</code> 41
	<code>Pstart</code> number can be printed in side 69	<code>\do@lineL</code> : Added <code>\do@lineLhook</code> to <code>\do@lineL</code> 39
v0.12	General: New new management of	Simplified <code>\do@lineL</code> by using macros for some common code 39
		<code>\do@lineR</code> : Changed <code>\do@lineR</code> similarly to <code>\do@lineL</code> 39
		<code>\do@lineRhook</code> : Added <code>\do@lineLhook</code> and <code>\do@lineRhook</code> 39
		<code>Leftside</code> : Added hooks into <code>Leftside</code> environment 34
		<code>\flag@end</code> : Removed extraneous spaces from <code>\flag@end</code> 29
		<code>\ifledRcol</code> : Moved <code>\ifl@dpairing</code> to <code>eledmac</code> 13
		<code>\ifpst@rtedR</code> : Moved <code>\ifpst@rtedL</code> to <code>eledmac</code> 14

<code>\l@dlinenumR:</code>	Simplified	v0.7	
<code>\leftlinenumR</code> and <code>\rightlinenumR</code>	by introducing <code>\l@dlinenumR</code>	19	General: Option ‘shiftedverses’ which make there is no blank between two parallel verses with inequal length. 1
<code>\l@dnumpstartsR:</code>	Moved		
<code>\l@dnumpstartsL</code> to <code>eledmac</code>		56	
<code>\ledsavedprintlines:</code>	Simplified	v0.8	
<code>\printlinesR</code> by using			General: Possibility to have a symbol on each hanging of verses, like in the french typography. Redefine the commande <code>\hangingsymbol</code> to define the character. 1
<code>\setprintlines</code>		48	
<code>\ledstrutR:</code> Added <code>\ledstrutL</code> and			
<code>\ledstrutR</code>		65	
<code>\normalbfnoteX:</code>	Removed		
extraneous spaces from			
<code>\normalbfnoteX</code>		52	
<code>\Pages:</code> Added <code>\ledstrutL</code> to		v0.9	General: Possibility to number <code>\pstart</code> 9
<code>\Pages</code>		64	Possibility to number the <code>pstart</code> with the commands <code>\numberpstarttrue</code> 1
Added <code>\ledstrutR</code> to <code>\Pages</code>		64	
<code>\Rightsidehookend:</code>	Added		
<code>\Leftsidehook</code> , <code>\Leftsidehookend</code> ,			<code>\ifledRcol:</code> Moved <code>\iflledRcol</code> and <code>\ifnumberingR</code> to <code>eledmac</code> 13
<code>\Rightsidehook</code> and <code>\Rightsidehookend</code>		34	
<code>\sublinenumrepR:</code>	Added	v0.9.1	
<code>\linenumrepR</code> and <code>\sublinenumrepR</code>			General: The numbering of the <code>pstarts</code> restarts on each <code>\beginnumbering</code> 1
		19	
v0.3a			
General: Minor <code>\linenummargin</code>		v0.9.2	General: Debug : with <code>\Columns</code> , the hanging indentation now runs on the left columns and the hanging symbol is shown only when <code>\stanza</code> is used. 1
fix			
		1	
<code>\line@marginR:</code>	Don’t just		
set <code>\line@marginR</code> in			
<code>\linenummargin</code>		17	
v0.3b			
General: Improved parallel page		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
balancing			
		1	
<code>\Pages:</code> Added <code>\l@dminpagelines</code>			
calculation for succeeding page			
pairs		65	
v0.3c			
General: Compatibilty with Poly-		v1.0	General: Compatibility with <code>eledmac</code> . Change name to <code>eledpar</code> . . 1
glossia			Debug in lineation by <code>pstart</code> . . 16
		1	
v0.4			
General: No more <code>ledparpatch</code> . All		v1.0.1	General: Correction on <code>\numberonlyfirstinline</code> with lineation by <code>pstart</code> or by page. 1
patches are now in the main			
file.		1	
v0.5			
General: Corrections about		v1.1	General: Shiftedverses becomes shiftedpstarts. 1
<code>\section</code> and other titles in			
numbered sections		1	
v0.6			
General: Be able to us <code>\chapter</code> in		v1.1.2	<code>\affixside@noteR:</code> Remove spurious space between line number
parallel pages.		1	

and line content	51	commands in parallel texts. . . .	1
v1.2			
General: Support for <code>\led{section}</code>			