

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 v1.4, last revised 2013/07/11.

[†]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	47
14.7 Penalties	47
14.8 Printing leftover notes	48
15 Footnotes	49
15.1 Normal footnote formatting	49
16 Cross referencing	49
17 Side notes	51
18 Familiar footnotes	53
19 Verse	53
20 Naming macros	55
21 Counts and boxes for parallel texts	56
22 Fixing babel	57
23 Parallel columns	59

<i>List of Figures</i>	3
24 Parallel pages	62
25 The End	71
References	72
Index	72
Change History	80

List of Figures

1 Introduction

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

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

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

This manual contains a general description of how to use **eledpar** starting in section 2; the complete source code for the package, with extensive documentation (in sections 8 through 25); and an Index to the source code. As **eledpar** is an adjunct to **eledmac** I assume that you have read the **eledmac** manual. Also **eledpar** requires **eledmac** to be used, preferably at least version 0.10 (2011/08/22). You do not need to read the source code for this package in order to use it but doing so may help to answer any questions you might have. On a first reading, I suggest that you should skip anything after the general documentation in sections 2 until 8, unless you are particularly interested in the innards of **eledpar**.

2 The **eledpar** package

A file may mix *numbered* and *unnumbered* text. Numbered text is printed with marginal line numbers and can include footnotes and endnotes that are referenced to those line numbers: this is how you'll want to print the text that you're editing. Unnumbered text is not printed with line numbers, and you can't use **eledmac**'s

note commands with it: this is appropriate for introductions and other material added by the editor around the edited text.

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

2.1 General

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

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

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

```
\maxchunks{5120}
```

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

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

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

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

3 Parallel columns

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

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

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

There is no required pagebreak before or after the columns.

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

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

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

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

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

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

4 Facing pages

pages Numbered text that is to be set on facing pages must be within a **pages** environment. Within the environment the text for the lefthand and righthand pages is placed within the **Leftside** and **Rightside** environments, respectively.

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

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

```

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

```

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

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

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

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

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

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

5 Left and right texts

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

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

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

\pstart In a serial (non-parallel) mode, each numbered paragraph, or chunk, is contained between the **\pstart** and **\pend** macros, and the paragraph is output when the **\pend** macro occurs. The situation is somewhat different with parallel typesetting as the left text (contained within **\pstart** and **\pend** groups within the

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

`Leftside` environment) has to be set in parallel with the right text (contained within its own `\pstart` and `\pend` groups within the corresponding `Rightside` environment) the `\pend` macros cannot immediately initiate any typesetting — this has to be controlled by the `\Columns` or `\Pages` macros. Several chunks may be specified within a `Leftside` or `Rightside` environment. A multi-chunk text then looks like:

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

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

Generally speaking, controls like `\firstlinenum` or `\linenummargin` apply to sequential and left texts. To effect right texts only they have to be within a `Rightside` environment.

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

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

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

6 Numbering text lines and paragraphs

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

lowed by `\endnumbering`, like:

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

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

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

`\memorydump`

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

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

`\Rlineflag`

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

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

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

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

`\printlinesR`
`\ledsavedprintlines`

The `\printlines` macro is ordinarily used to print the line number references for critical footnotes. For footnotes from right side texts a special version is supplied, called `\printlinesR`, which incorporates `\Rlineflag`. (The macro `\ledsavedprintlines` is a copy of the original `\printlines`, just in case ...).

As provided, the package makes no use of `\printlinesR` but you may find it useful. For example, if you only use the B footnote series in righthand texts then you may wish to flag any line numbers in those footnotes with the value of `\Rlineflag`. You could do this by putting the following code in your preamble:

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

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

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

7 Verse

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

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

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

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

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

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

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

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

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

\end{astanza}
...
```

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

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

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

\end{astanza}
...
```

`\hangingsymbol`

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

character in each hanged line. If you use it, you must run \LaTeX two time. Example for the french typographie

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

8 Implementation overview

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

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

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

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

9 Preliminaries

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

```

1 (*code)
2 \NeedsTeXFormat{LaTeX2e}
3 \ProvidesPackage{eledpar}[2013/07/11 v1.4 eledmac extension for parallel texts]
4
```

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

```
\ifshiftedpstarts
```

```

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

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

hidden from the user by letting the `Leftside` and `Rightside` environments set things up appropriately.

```
\ifl@dpairing \ifl@dpairing is set TRUE if we are processing parallel texts and \ifl@dpaging
\ifl@dpaging is also set TRUE if we are doing parallel pages. \ifledRcol is set TRUE if we
\ifledRcol are doing the right hand text. \ifl@dpairing is defined in eledmac.
11 \l@dpairingfalse
12 \newif\ifl@dpaging
13 \l@dpagingfalse
14 \ledRcolfalse

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

9.1 Messages

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

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

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

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

10 Sectioning commands

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

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

```
30 \newcount\section@numR
31 \section@numR=\z@
```

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

```

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

```

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

```

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

```

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

```

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

```

73

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

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

```

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

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

```

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

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

```

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

```

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

```

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

```

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

```

11 Line counting

11.1 Choosing the system of lineation

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

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

<code>\ifbypstart@R</code>	
<code>\bypstart@Rtrue</code>	• line-of-page : <code>bypstart@R = false</code> and <code>bypage@R = true</code> .
<code>\bypstart@Rfalse</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 `\insert@countR` two arguments:

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

```

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     \@ifundefined{xpg@main@language}{%if not polyglossia
664       \flag@start}%
665       {\if@RTL\flag@end\else\flag@start\fi% be careful on the direction of writing with p
666     }%
667   \endgroup
668   \showlemma{#1}%
669   \ifx\end@lemmas\empty \else
670     \gl@p\end@lemmas\to\x@lemma
671     \x@lemma
672     \global\let\x@lemma=\relax
673   \fi
674   \@ifundefined{xpg@main@language}{%if not polyglossia
675     \flag@end}%
676     {\if@RTL\flag@start\else\flag@end\fi% be careful on the direction of writing with p
677   }
678 }
```

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

```

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

```



```

700     }%
701 }
702

```

`\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`.

```

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

```

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 724 \newenvironment{pairs}{%}
725   \l@dpairingtrue
726   \l@dpagingfalse
727 }{%
728   \l@dpairingfalse
729 }

```

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.

```

730 \newenvironment{pages}{%
731   \let\oldchapter\chapter
732   \let\chapter\chapterinpages

```

```

733 \l@dpairingtrue
734 \l@dpagingtrue
735 \setlength{\Lcolwidth}{\textwidth}%
736 \setlength{\Rcolwidth}{\textwidth}%
737 }{%
738 \l@dpairingfalse
739 \l@dpagingfalse
740 \let\chapter\oldchapter
741 }
742 \newcommand{\chapterinpages}{\thispagestyle{plain}%
743 \global\@topnum\z@
744 \@afterindentfalse
745 \secdef\@chapter\@schapter}
746

```

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

```

747 \newif\ifinstanzaL
748 \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.

```

749 \newenvironment{Leftside}{%
750 \ledRcolfalse
751 \let\beginnumbering\beginnumbering\setcounter{pstartL}{1}
752 \let\pstart\pstartL
753 \let\thepstart\thepstartL
754 \let\pend\pendL
755 \let\memorydump\memorydumpL
756 \Leftsidehook
757 \let\oldstanza\stanza
758 \renewcommand{\stanza}{\oldstanza\global\instanzaLtrue}
759 }{
760 \let\stanza\oldstanza
761 \Leftsidehookend}

```

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

```

\Leftsidehookend
\Rightsidehook 762 \newcommand*{\Leftsidehook}{}
\Rightsidehookend 763 \newcommand*{\Leftsidehookend}{}
764 \newcommand*{\Rightsidehook}{}
765 \newcommand*{\Rightsidehookend}{}
766

```

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.

```

767 \newenvironment{Rightside}{%
768   \ledRcoltrue
769   \let\beginnumbering\beginnumberingR
770   \let\endnumbering\endnumberingR
771   \let\pausenumbering\pausenumberingR
772   \let\resumenumbering\resumenumberingR
773   \let\memorydump\memorydumpR
774   \let\thepstart\thepstartR
775   \let\pstart\pstartR
776   \let\pend\pendR
777   \let\lineation\lineationR
778   \Rightsidehook
779   \let\oldstanza\stanza
780   \renewcommand{\stanza}{\oldstanza\global\instanzaRtrue}
781 }{%
782   \ledRcolfalse
783   \let\stanza\oldstanza
784   \Rightsidehookend
785 }
786

```

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.

```

787 \newcount\num@linesR
788 \newbox\one@lineR
789 \newcount\par@lineR

```

\pstartL `changesv1.12012/09/25Add \labelpstarttrue (from eledmac). changesv1.1.12012/10/01Correct`
\pstartR `\pstartR` bug introduced by 1.1. `\pstart` starts the paragraph by clearing the

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

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

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

```

790
791 \newcounter{pstartL}
792 \newcounter{pstartLold}
793 \renewcommand{\thepstartL}{\bfseries\@arabic\c@pstartL}. }
794 \newcounter{pstartR}
795 \newcounter{pstartRold}
796 \renewcommand{\thepstartR}{\bfseries\@arabic\c@pstartR}. }
797
798 \newcommand*{\pstartL}{
799 \if@nobreak
800   \let\@oldnobreak\@nobreaktrue
801 \else
802   \let\@oldnobreak\@nobreakfalse
803 \fi
804   \@nobreaktrue
805   \ifnumbering \else
806     \led@err@PstartNotNumbered
807     \beginnumbering
808   \fi
809   \ifnumberedpar@
810     \led@err@PstartInPstart
811   \pend
812 \fi

```

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

```

813 \ifpst@rtedL\else
814   \setcounter{pstartLold}{\value{pstartL}}%
815   \list@clear{\inserts@list}%
816   \global\let\next@insert=\empty
817   \global\pst@rtedLtrue
818 \fi
819 \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.

```

820 \global\advance\l@dnumpstartsL \@ne
821 \ifnum\l@dnumpstartsL>\l@dc@maxchunks
822   \led@err@TooManyPstarts

```

```

823   \global\l@dnumpstartsL=\l@dc@maxchunks
824   \fi
825   \global\setnamebox{l@dLcolrawbox\the\l@dnumpstartsL}=\vbox\bgroup\ifautopar\else\ifnumberpstart\ifs
826   \hsize=\Lcolwidth
827   \numberedpar@true
828   \iflabelpstart\protected@edef\@currentlabel
829   {\p@pstartL\thepstartL}\fi
830   }

831 \newcommand*{\pstartR}{
832 \if@nobreak
833   \let\@oldnobreak\@nobreaktrue
834 \else
835   \let\@oldnobreak\@nobreakfalse
836 \fi
837   \@nobreaktrue
838   \ifnumberingR \else
839   \led@err@PstartNotNumbered
840   \beginnumberingR
841   \fi
842   \ifnumberedpar@
843   \led@err@PstartInPstart
844   \pendR
845   \fi
846   \ifpst@rtedR\else
847   \setcounter{pstartRold}{\value{pstartR}}%
848   \list@clear{\inserts@listR}%
849   \global\let\next@insertR=\empty
850   \global\pst@rtedRtrue
851   \fi
852   \begingroup\normal@pars
853   \global\advance\l@dnumpstartsR \@ne
854   \ifnum\l@dnumpstartsR>\l@dc@maxchunks
855   \led@err@TooManyPstarts
856   \global\l@dnumpstartsR=\l@dc@maxchunks
857   \fi
858   \global\setnamebox{l@dRcolrawbox\the\l@dnumpstartsR}=\vbox\bgroup\ifautopar\else\ifnumberpstart\ifs
859   \hsize=\Rcolwidth
860   \numberedpar@true
861   \iflabelpstart\protected@edef\@currentlabel
862   {\p@pstartR\thepstartR}\fi
863   }

```

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

```

864 \newcommand*{\pendL}{\ifnumbering \else
865   \led@err@PendNotNumbered
866   \fi
867   \ifnumberedpar@ \else
868   \led@err@PendNoPstart

```

```
869 \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.

```
870 \l@dzeroopenalties
871 \endgraf\global\num@lines=\prevgraf\egroup
872 \global\par@line=0
```

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

```
873 \endgroup
874 \ignorespaces
875 \@oldnobreak
876 \ifnumberpstart
877 \addtocounter{pstartL}{1}
878 \fi}
879
```

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

```
880 \newcommand*{\pendR}{\ifnumberingR \else
881 \led@err@PendNotNumbered
882 \fi
883 \ifnumberedpar@ \else
884 \led@err@PendNoPstart
885 \fi
886 \l@dzeroopenalties
887 \endgraf\global\num@linesR=\prevgraf\egroup
888 \global\par@lineR=0
889 \endgroup
890 \ignorespaces
891 \@oldnobreak
892 \ifnumberpstart
893 \addtocounter{pstartR}{1}
894 \fi
895 }
896
```

14.2 Processing one line

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

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

```
897 \newbox\l@dleftbox
898 \newbox\l@drightbox
899
```

```

\countLline We need to know the number of lines processed.
\countRline 900 \newcount\countLline
              901 \countLline \z@
              902 \newcount\countRline
              903 \countRline \z@
              904

\@donereallinesL We need to know the number of ‘real’ lines output (i.e., those that have been input
\@donetotallinesL by the user), and the total lines output (which includes any blank lines output for
\@donereallinesR synchronisation).
\@donetotallinesR 905 \newcount\@donereallinesL
                  906 \newcount\@donetotallinesL
                  907 \newcount\@donereallinesR
                  908 \newcount\@donetotallinesR
                  909

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

          910 \newcommand*{\do@lineL}{%
          911 \advance\countLline \@ne
          912 \ifvbox\namebox{\l@dLcolrawbox\the\l@dpscl}%
          913 {\vbadness=10000
          914 \splittopskip=\z@
          915 \do@lineLhook
          916 \l@demptyd@ta
          917 \global\setbox\one@line=\vsplit\namebox{\l@dLcolrawbox\the\l@dpscl}
          918 to\baselineskip}%
          919 \unvbox\one@line \global\setbox\one@line=\lastbox
          920 \getline@numL
          921 \ifnum\@lock>\@ne\inserthangingsymboltrue\else\inserthangingsymbolfalse\fi
          922 \setbox\l@dleftbox
          923 \hb@xt@ \Lcolwidth{%
          924 \affixpstart@numL
          925 \affixline@num
          926 \l@dld@ta
          927 \add@inserts
          928 \affixside@note
          929 \l@dlsn@te
          930 {\ledllfill\hb@xt@ \wd\one@line{\do@insidelineLhook\inserthangingsymbolL\new@line\l@dunhbox@linef
          931 \l@drsn@te
          932 }}%
          933 \add@penaltiesL
          934 \global\advance\@donereallinesL\@ne
          935 \global\advance\@donetotallinesL\@ne
          936 \else
          937 \setbox\l@dleftbox \hb@xt@ \Lcolwidth{\hspace*{\Lcolwidth}}%
          938 \global\advance\@donetotallinesL\@ne
          939 \fi}

```

940

941

```

\do@lineLhook Hooks, initially empty, into the respective \do@line(L/R) macros.
\do@lineRhook 942 \newcommand*{\do@lineLhook}{}
\do@insidelineLhook 943 \newcommand*{\do@lineRhook}{}
\do@insidelineRhook 944 \newcommand*{\do@insidelineLhook}{}
945 \newcommand*{\do@insidelineRhook}{}
946

```

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

```

947 \newcommand*{\do@lineR}{%
948   \advance\countRline \@ne
949   \ifvbox\namebox{\l@dRcolrawbox\the\l@dpscR}%
950   {\vbadness=10000
951     \splittopskip=\z@
952     \do@lineRhook
953     \l@emptyd@ta
954     \global\setbox\one@lineR=\vsplit\namebox{\l@dRcolrawbox\the\l@dpscR}
955     to\baselineskip}%
956   \unvbox\one@lineR \global\setbox\one@lineR=\lastbox
957   \getline@numR
958   \ifnum\@lockR>\@ne\inserthangingsymbolRtrue\else\inserthangingsymbolRfalse\fi
959   \setbox\l@dtrightbox
960   \hb@xt@ \Rcolwidth{%
961     \affixpstart@numR
962     \affixline@numR
963     \l@dld@ta
964     \add@insertsR
965     \affixside@noteR
966     \l@dlsn@te
967     {\correcthangingsR\ledllfill\hb@xt@ \wd\one@lineR{\do@insidelineRhook\inserthangingsym
968     \l@drrsn@te
969   }}%
970   \add@penaltiesR
971   \global\advance\@donereallinesR\@ne
972   \global\advance\@donetotallinesR\@ne
973 \else
974   \setbox\l@dtrightbox \hb@xt@ \Rcolwidth{\hspace*{\Rcolwidth}}
975   \global\advance\@donetotallinesR\@ne
976 \fi}
977
978

```

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.


```

979 \newcommand*{\getline@numR}{%
980   \global\advance\absline@numR \@ne
981   \do@actionsR
982   \do@ballastR
983 \ifnumberline
984   \ifsublines@
985     \ifnum\sub@lockR<\tw@
986       \global\advance\subline@numR \@ne
987     \fi
988   \else
989     \ifnum\@lockR<\tw@
990       \global\advance\line@numR \@ne
991       \global\subline@numR \z@
992     \fi
993   \fi
994 \fi
995 }
996 \newcommand*{\getline@numL}{%
997   \global\advance\absline@num \@ne
998   \do@actions
999   \do@ballast
1000 \ifnumberline
1001   \ifsublines@
1002     \ifnum\sub@lock<\tw@
1003       \global\advance\subline@num \@ne
1004     \fi
1005   \else
1006     \ifnum\@lock<\tw@
1007       \global\advance\line@num \@ne
1008       \global\subline@num \z@
1009     \fi
1010   \fi
1011 \fi
1012 }
1013
1014

```

`\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.

```

1015 \newcommand*{\do@ballastR}{\global\ballast@count=\z@
1016   \begingroup
1017     \advance\absline@numR \@ne
1018     \ifnum\next@actionlineR=\absline@numR
1019       \ifnum\next@actionR>-1001
1020         \global\advance\ballast@count by -\c@ballast
1021       \fi
1022     \fi
1023   \endgroup}

```

`\do@actionsR` The `\do@actionsR` macro looks at the list of actions to take at particular right

```

\do@actions@fixedcodeR
\do@actions@nextR

```

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.

```

1024 \newcommand*{\do@actions@fixedcodeR}{%
1025   \ifcase\@l@dtmpcnta%
1026   \or% % 1001
1027     \global\sublines@true
1028   \or% % 1002
1029     \global\sublines@false
1030   \or% % 1003
1031     \global\@lockR=\@ne
1032   \or% % 1004
1033     \ifnum\@lockR=\tw@
1034       \global\@lockR=\thr@@
1035     \else
1036       \global\@lockR=\z@
1037     \fi
1038   \or% % 1005
1039     \global\sub@lockR=\@ne
1040   \or% % 1006
1041     \ifnum\sub@lockR=\tw@
1042       \global\sub@lockR=\thr@@
1043     \else
1044       \global\sub@lockR=\z@
1045     \fi
1046   \or% % 1007
1047     \l@dskipnumbertrue
1048   \else
1049     \led@warn@BadAction
1050   \fi}
1051
1052
1053 \newcommand*{\do@actionsR}{%
1054   \global\let\do@actions@nextR=\relax
1055   \@l@dtmpcntb=\absline@numR
1056   \ifnum\@l@dtmpcntb<\next@actionlineR\else
1057     \ifnum\next@actionR>-1001\relax
1058       \global\page@numR=\next@actionR
1059       \ifbypage@R
1060         \global\line@numR \z@ \global\subline@numR \z@
1061       \fi
1062     \else
1063       \ifnum\next@actionR<-4999\relax % 9/05 added relax here
1064         \@l@dtmpcnta=-\next@actionR
1065         \advance\@l@dtmpcnta by -5001\relax
1066         \ifsublines@
1067           \global\subline@numR=\@l@dtmpcnta
1068         \else

```

```

1069      \global\line@numR=\@l@tempcnta
1070      \fi
1071      \else
1072          \@l@tempcnta=-\next@actionR
1073          \advance\@l@tempcnta by -1000\relax
1074          \do@actions@fixedcodeR
1075      \fi
1076  \fi
1077  \ifx\actionlines@listR\empty
1078      \gdef\next@actionlineR{1000000}%
1079  \else
1080      \gl@p\actionlines@listR\to\next@actionlineR
1081      \gl@p\actions@listR\to\next@actionR
1082      \global\let\do@actions@nextR=\do@actionsR
1083  \fi
1084  \fi
1085  \do@actions@nextR}
1086

```

14.4 Line number printing

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

```

\ch@cksub@l@ckR 1087
\ch@ck@l@ckR 1088 \providecommand*\l@dcalcnum}[3]{%
\fx@l@cksR 1089 \ifnum #1 > #2\relax
\affixline@numR 1090 \@l@tempcnta = #1\relax
1091 \advance\@l@tempcnta by -#2\relax
1092 \divide\@l@tempcnta by #3\relax
1093 \multiply\@l@tempcnta by #3\relax
1094 \advance\@l@tempcnta by #2\relax
1095 \else
1096 \@l@tempcnta=#2\relax
1097 \fi}
1098
1099 \newcommand*\ch@cksub@l@ckR}{%
1100 \ifcase\sub@lockR
1101 \or
1102 \ifnum\sublock@disp=\@ne
1103 \@l@tempcntb \z@ \@l@tempcnta \@ne
1104 \fi
1105 \or
1106 \ifnum\sublock@disp=\tw@
1107 \else
1108 \@l@tempcntb \z@ \@l@tempcnta \@ne
1109 \fi
1110 \or
1111 \ifnum\sublock@disp=\z@
1112 \@l@tempcntb \z@ \@l@tempcnta \@ne
1113 \fi

```

```

1114 \fi}
1115
1116 \newcommand*{\ch@ck@l@ckR}{%
1117 \ifcase\@lockR
1118 \or
1119 \ifnum\lock@disp=\@ne
1120 \l@dttempcntb \z@ \l@dttempcnta \@ne
1121 \fi
1122 \or
1123 \ifnum\lock@disp=\tw@
1124 \else
1125 \l@dttempcntb \z@ \l@dttempcnta \@ne
1126 \fi
1127 \or
1128 \ifnum\lock@disp=\z@
1129 \l@dttempcntb \z@ \l@dttempcnta \@ne
1130 \fi
1131 \fi}
1132
1133 \newcommand*{\f@x@l@cksR}{%
1134 \ifcase\@lockR
1135 \or
1136 \global\@lockR \tw@
1137 \or \or
1138 \global\@lockR \z@
1139 \fi
1140 \ifcase\sub@lockR
1141 \or
1142 \global\sub@lockR \tw@
1143 \or \or
1144 \global\sub@lockR \z@
1145 \fi}
1146
1147
1148 \newcommand*{\affixline@numR}{%
1149 \ifnumberline
1150 \ifl@dskipnumber
1151 \global\l@dskipnumberfalse
1152 \else
1153 \ifsublines@
1154 \l@dttempcntb=\subline@numR
1155 \l@dcalcnm{\subline@numR}{\c@firstsublinenumR}{\c@sublinenumincrementR}%
1156 \ch@cksub@lockR
1157 \else
1158 \l@dttempcntb=\line@numR
1159 \ifx\linenumberlist\empty
1160 \l@dcalcnm{\line@numR}{\c@firstlinenumR}{\c@linenumincrementR}%
1161 \else
1162 \l@dttempcnta=\line@numR
1163 \edef\rem@inder{\linenumberlist,\number\line@numR,}%

```

```

1164      \edef\sc@n@list{\def\noexpand\sc@n@list
1165        #####1,\number\@l@tempcnta,####2|{\def\noexpand\rem@inder{####2}}}%
1166      \sc@n@list\expandafter\sc@n@list\rem@inder|%
1167      \ifx\rem@inder\empty\advance\@l@tempcnta\@ne\fi
1168      \fi
1169      \ch@ck@l@ckR
1170    \fi
1171    \ifnum\@l@tempcnta=\@l@tempcntb
1172      \if@twocolumn
1173        \if@firstcolumn
1174          \gdef\l@dld@ta{\llap{\leftlinenumR}}}%
1175        \else
1176          \gdef\l@drd@ta{\rlap{\rightlinenumR}}}%
1177        \fi
1178      \else
1179        \@l@tempcntb=\line@marginR
1180        \ifnum\@l@tempcntb>\@ne
1181          \advance\@l@tempcntb by\page@numR
1182        \fi
1183        \ifodd\@l@tempcntb
1184          \gdef\l@drd@ta{\rlap{\rightlinenumR}}}%
1185        \else
1186          \gdef\l@dld@ta{\llap{\leftlinenumR}}}%
1187        \fi
1188      \fi
1189    \fi
1190    \f@x@l@cksR
1191  \fi
1192 \fi}

```

14.5 Pstart number printing in side

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

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

```

\affixpstart@numL
\affixpstart@numR 1193
  \leftpstartnumR 1194 \newcommand*{\affixpstart@numL}{%
\rightpstartnumR 1195 \ifsidepstartnum
\leftpstartnumL 1196 \if@twocolumn
\rightpstartnumL 1197   \if@firstcolumn
\ifpstartnumR 1198     \gdef\l@dld@ta{\llap{\leftpstartnumL}}}%
1199   \else
1200     \gdef\l@drd@ta{\rlap{\rightpstartnumL}}}%
1201   \fi

```

```

1202 \else
1203 \l@dttempcntb=\line@margin
1204 \ifnum\l@dttempcntb>\@ne
1205 \advance\l@dttempcntb \page@num
1206 \fi
1207 \ifodd\l@dttempcntb
1208 \gdef\l@drd@ta{\rlap{\rightpstartnumL}}}%
1209 \else
1210 \gdef\l@dld@ta{\llap{\leftpstartnumL}}}%
1211 \fi
1212 \fi
1213 \fi
1214 }
1215 \newcommand*\affixpstart@numR{%
1216 \ifsidepstartnum
1217 \if@twocolumn
1218 \if@firstcolumn
1219 \gdef\l@dld@ta{\llap{\leftpstartnumR}}}%
1220 \else
1221 \gdef\l@drd@ta{\rlap{\rightpstartnumR}}}%
1222 \fi
1223 \else
1224 \l@dttempcntb=\line@marginR
1225 \ifnum\l@dttempcntb>\@ne
1226 \advance\l@dttempcntb \page@numR
1227 \fi
1228 \ifodd\l@dttempcntb
1229 \gdef\l@drd@ta{\rlap{\rightpstartnumR}}}%
1230 \else
1231 \gdef\l@dld@ta{\llap{\leftpstartnumR}}}%
1232 \fi
1233 \fi
1234 \fi
1235 }
1236
1237 \newcommand*\leftpstartnumL){
1238 \ifpstartnum
1239 \thepstartL
1240 \kern\linenumsep\global\pstartnumfalse\fi
1241 }
1242 \newcommand*\rightpstartnumL){
1243 \ifpstartnum\kern\linenumsep
1244 \thepstartL
1245 \global\pstartnumfalse\fi
1246 }
1247 \newif\ifpstartnumR
1248 \pstartnumRtrue
1249 \newcommand*\leftpstartnumR){
1250 \ifpstartnumR
1251 \thepstartR

```

```

1252 \kern\linenumsep\global\pstartnumRfalse\fi
1253 }
1254 \newcommand*{\rightpstartnumR}{
1255 \ifpstartnumR\kern\linenumsep
1256 \thepstartR
1257 \global\pstartnumRfalse\fi
1258 }

```

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.

```
1259 \list@create{\inserts@listR}
```

`\add@insertsR` The right text version.

```

\add@inserts@nextR 1260 \newcommand*{\add@insertsR}{%
1261 \global\let\add@inserts@nextR=\relax
1262 \ifx\inserts@listR\empty \else
1263 \ifx\next@insertR\empty
1264 \ifx\insertlines@listR\empty
1265 \global\noteschanged@true
1266 \gdef\next@insertR{100000}%
1267 \else
1268 \gl@p\insertlines@listR\to\next@insertR
1269 \fi
1270 \fi
1271 \ifnum\next@insertR=\absline@numR
1272 \gl@p\inserts@listR\to\@insertR
1273 \@insertR
1274 \global\let\@insertR=\undefined
1275 \global\let\next@insertR=\empty
1276 \global\let\add@inserts@nextR=\add@insertsR
1277 \fi
1278 \fi
1279 \add@inserts@nextR}
1280

```

14.7 Penalties

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

In the code below, which is a virtual copy of the original `\add@penalties`, `\num@lines` is the number of lines in the whole paragraph, and `\par@line` is the line we're working on at the moment. The count `\@l@tempcnta` is used to calculate and accumulate the penalty; it is initially set to the value of `\ballast@count`,

which has been worked out in `\do@ballast`. Finally, the penalty is checked to see that it doesn't go below -10000 .

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

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

```
1281 \newcommand*{\add@penaltiesL}{\}
1282 \newcommand*{\add@penaltiesR}{\}
1283
```

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.

```
1284 \newcommand*{\flush@notesR}{\%
1285 \xloop
1286 \ifx\inserts@listR\empty \else
1287 \glp\inserts@listR\to\@insertR
1288 \@insertR
1289 \global\let\@insertR=\undefined
1290 \repeat}
1291
```


15 Footnotes

15.1 Normal footnote formatting

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

```

\printlinesR This is the right text version of \printlines and takes account of \Rlineflag.
\ledsavedprintlines Just in case, \ledsavedprintlines is a copy of the original \printlines.
    Just a reminder of the arguments:
\printlinesR      #1      | #2 | #3 | #4 | #5 | #6 | #7
\printlinesR start-page | line | subline | end-page | line | subline | font
1292 \def\printlinesR#1|#2|#3|#4|#5|#6|#7|{\begingroup
1293   \setprintlines{#1}{#2}{#3}{#4}{#5}{#6}%
1294   \ifl@d@pnum #1\fullstop\fi
1295   \ifledplinenum \linenumr@p{#2}\Rlineflag\else \sympninenum\fi
1296   \ifl@d@ssub \fullstop \sublinenumr@p{#3}\fi
1297   \ifl@d@dash \endashchar\fi
1298   \ifl@d@pnum #4\fullstop\fi
1299   \ifl@d@elin \linenumr@p{#5}\Rlineflag\fi
1300   \ifl@d@esl \ifl@d@elin \fullstop\fi \sublinenumr@p{#6}\fi
1301 \endgroup}
1302
1303 \let\ledsavedprintlines\printlines
1304

```

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.

```

1305 \list@create{\labelref@listR}
1306

```

`\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.

```

1307 \renewcommand*{\edlabel}[1]{\@bsphack
1308   \ifledRcol
1309     \write\linenum@outR{\string\@lab}%
1310     \ifx\labelref@listR\empty
1311       \xdef\label@refs{\zz@@@}%
1312     \else

```

```

1313     \glp\labelref@listR\to\label@refs
1314     \fi
1315     \ifvmode
1316         \advancelabel@refs
1317     \fi
1318     \protected@write\@auxout{%
1319         {\string\l@dmake@labelsR\space\thepage|\label@refs|{#1}}%
1320     \else
1321         \write\linenum@out{\string\@lab}%
1322         \ifx\labelref@list\empty
1323             \xdef\label@refs{\zz@@}%
1324         \else
1325             \glp\labelref@list\to\label@refs
1326         \fi
1327         \ifvmode
1328             \advancelabel@refs
1329         \fi
1330     \protected@write\@auxout{%
1331         {\string\l@dmake@labelsR\space\thepage|\label@refs|{#1}}%
1332     \fi
1333     \@esphack}
1334

```

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

```

1335 \def\l@dmake@labelsR#1|#2|#3|#4{%
1336     \expandafter\ifx\csname the@label#4\endcsname \relax\else
1337         \led@warn@DuplicateLabel{#4}%
1338     \fi
1339     \expandafter\gdef\csname the@label#4\endcsname{#1|#2\Rlineflag|#3}%
1340     \ignorespaces}
1341 \AtBeginDocument{%
1342     \def\l@dmake@labelsR#1|#2|#3|#4{%
1343     }
1344

```

`\@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.

```

1345 \renewcommand*{\@lab}{%
1346     \ifledRcol
1347         \xright@appenditem{\linenumr@p{\line@numR}}|{%
1348             \ifsublines@ \sublinenumr@p{\subline@numR}\else 0\fi}%
1349         \to\labelref@listR
1350     \else
1351         \xright@appenditem{\linenumr@p{\line@num}}|{%
1352             \ifsublines@ \sublinenumr@p{\subline@num}\else 0\fi}%
1353         \to\labelref@list
1354     \fi}
1355

```

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 1356 \newcount\sidenote@marginR
                  1357 \renewcommand*{\sidenotemargin}[1]{\%
                  1358 \l@dgetsidenote@margin{#1}%
                  1359 \ifnum\@l@dttempcntb>\m@ne
                  1360 \ifledRcol
                  1361 \global\sidenote@marginR=\@l@dttempcntb
                  1362 \else
                  1363 \global\sidenote@margin=\@l@dttempcntb
                  1364 \fi
                  1365 \fi}}
                  1366 \sidenotemargin{right}
                  1367 \global\sidenote@margin=\@ne
                  1368
```

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

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

```
\l@dcsnote 1369 \renewcommand*{\l@dlsnote}[1]{\%
            1370 \begingroup%
            1371 \newcommand{\content}{#1}%
            1372 \ifnumberedpar@
            1373 \ifledRcol%
            1374 \xright@appenditem{\noexpand\l@dlsnote{\csexpandonce{content}}}%
            1375 \to\inserts@listR
            1376 \global\advance\insert@countR \@ne%
            1377 \else%
            1378 \xright@appenditem{\noexpand\l@dlsnote{\csexpandonce{content}}}%
            1379 \to\inserts@list
            1380 \global\advance\insert@count \@ne%
            1381 \fi
            1382 \fi\ignorespaces\endgroup}
            1383 \renewcommand*{\l@drsnote}[1]{\%
            1384 \begingroup%
            1385 \newcommand{\content}{#1}%
            1386 \ifnumberedpar@
            1387 \ifledRcol%
            1388 \xright@appenditem{\noexpand\l@drsnote{\csexpandonce{content}}}%
            1389 \to\inserts@listR
            1390 \global\advance\insert@countR \@ne%
            1391 \else%
            1392 \xright@appenditem{\noexpand\l@drsnote{\csexpandonce{content}}}%
            1393 \to\inserts@list
            1394 \global\advance\insert@count \@ne%
            1395 \fi
            1396 \fi\ignorespaces\endgroup}
```

```

1397 \renewcommand*{\l@dcnote}[1]{%
1398   \begingroup%
1399   \newcommand{\content}{#1}%
1400   \ifnumberedpar@
1401     \ifledRcol%
1402       \xright@appenditem{\noexpand\l@dcnote{\csexpandonce{content}}}%
1403         \to\inserts@listR
1404       \global\advance\insert@countR \@ne%
1405     \else%
1406       \xright@appenditem{\noexpand\l@dcnote{\csexpandonce{content}}}%
1407         \to\inserts@list
1408       \global\advance\insert@count \@ne%
1409     \fi
1410   \fi\ignorespaces\endgroup}
1411

```

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

```

1412 \newcommand*{\affixside@noteR}{%
1413   \def\sidenotecontent@{}%
1414   \numdef{\itemcount@}{0}%
1415   \def\do##1{%
1416     \ifnumequal{\itemcount@}{0}%
1417       {%
1418         \appto\sidenotecontent@{##1}% Not print not separator before the 1st note
1419         {\appto\sidenotecontent@{\sidenotesep ##1}%
1420         }%
1421         \numdef{\itemcount@}{\itemcount@+1}%
1422       }%
1423   \dolistloop{\l@dcnotetext}%
1424   \ifnumgreater{\itemcount@}{1}{\eledmac@warning{\itemcount@\space sidenotes on line \tl
1425   \gdef\@templ@d}{%
1426   \ifx\@templ@d\l@dcnotetext \else%
1427     \if@twocolumn%
1428       \if@firstcolumn%
1429         \setl@dlp@rbox{\sidenotecontent@}%
1430       \else%
1431         \setl@drp@rbox{\sidenotecontent@}%
1432       \fi%
1433     \else%
1434       \@l@tempcntb=\sidenote@marginR%
1435       \ifnum\@l@tempcntb>\@ne%
1436         \advance\@l@tempcntb by\page@num%
1437       \fi%
1438       \ifodd\@l@tempcntb%
1439         \setl@drp@rbox{\sidenotecontent@t}%
1440       \else%
1441         \setl@dlp@rbox{\sidenotecontent@}%
1442       \fi%
1443     \fi%
1444   \fi}

```

1445

18 Familiar footnotes

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

```

1446 \renewcommand{\l@dbfnote}[1]{%
1447   \ifnumberedpar@
1448   \gdef\@tag{#1}%
1449   \ifledRcol%
1450     \xright@appenditem{\noexpand\vl@dbfnote{\csexpandonce{\@tag}}{\@thefnmark}}%
1451                       \to\inserts@listR
1452     \global\advance\insert@countR \@ne%
1453   \else%
1454     \xright@appenditem{\noexpand\vl@dbfnote{\csexpandonce{\@tag}}{\@thefnmark}}%
1455                       \to\inserts@list
1456     \global\advance\insert@count \@ne%
1457   \fi
1458 \fi\ignorespaces}
1459
```

`\normalbfnoteX`

```

1460 \renewcommand{\normalbfnoteX}[2]{%
1461   \ifnumberedpar@
1462   \ifledRcol%
1463     \protected@csxdef{thisfootnote}{\csuse{thefootnote#1}}%
1464     \xright@appenditem{\noexpand\vbfnoteX{#1}{#2}{\csexpandonce{thisfootnote}}}%
1465                       \to\inserts@listR
1466     \global\advance\insert@countR \@ne%
1467   \else%
1468     \protected@csxdef{thisfootnote}{\csuse{thefootnote#1}}%
1469     \xright@appenditem{\noexpand\vbfnoteX{#1}{#2}{\csexpandonce{thisfootnote}}}%
1470                       \to\inserts@list
1471     \global\advance\insert@count \@ne%
1472   \fi
1473 \fi\ignorespaces}
1474
```

19 Verse

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

`\inserthangingsymbolL`

```

\inserthangingsymbolR 1475 \newif\ifinserthangingsymbolR
1476 \newcommand{\inserthangingsymbolL}{%
1477 \ifinserthangingsymbol%
```

```

1478     \ifistanzaL%
1479         \hfill\hangingsymbol%
1480     \fi%
1481 \fi}
1482 \newcommand{\inserthangingsymbolR}{%
1483 \ifinserthangingsymbolR%
1484     \ifistanzaR%
1485         \hfill\hangingsymbol%
1486     \fi%
1487 \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 1488 \newcommand{\correcthangingL}{%
1489 \ifl@dpaging\else%
1490     \ifistanzaL%
1491         \ifinserthangingsymbol%
1492             \hskip \@ifundefined{sza@00}{0}{\expandafter%
1493                 \noexpand\csname sza@00\endcsname}\stanzaindentbase%
1494         \fi%
1495     \fi%
1496 \fi}
1497
1498 \newcommand{\correcthangingR}{%
1499 \ifl@dpaging\else%
1500     \ifistanzaR%
1501         \ifinserthangingsymbolR%
1502             \hskip \@ifundefined{sza@00}{0}{\expandafter%
1503                 \noexpand\csname sza@00\endcsname}\stanzaindentbase%
1504         \fi%
1505     \fi%
1506 \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.

```

1507 \chardef\next=\catcode'\&
1508 \catcode'\&=\active
1509

```

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

```

1510 \newenvironment{astanza}{%
1511     \startstanzahook
1512     \catcode'\&\active

```

```

1513 \global\stanza@count\@ne
1514 \ifnum\usernamecount{sza@00}=\z@
1515   \let\stanza@hang\relax
1516   \let\endlock\relax
1517 \else
1518 %%    \interlinepenalty\@M % this screws things up, but I don't know why
1519   \rightskip\z@ plus 1fil\relax
1520 \fi
1521 \ifnum\usernamecount{szp@00}=\z@
1522   \let\sza@penalty\relax
1523 \fi
1524 \def&{%
1525   \endlock\mbox{}}%
1526   \sza@penalty
1527   \global\advance\stanza@count\@ne
1528   \@astanza@line}%
1529 \def\&{%
1530   \endlock\mbox{}}
1531   \pend
1532   \endstanzaextra}%
1533 \pstart
1534 \@astanza@line
1535 }{}
1536

```

`\@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.

```

1537 \newcommand*{\@astanza@line}{%
1538   \parindent=\csname sza@\number\stanza@count @\endcsname\stanzaindentbase
1539   \par
1540   \stanza@hang%\mbox{}}%
1541   \ignorespaces}
1542

```

Lastly reset the modified category codes.

```

1543 \catcode'\&=\next
1544

```

20 Naming macros

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

```

\newnamebox A set of macros for creating and using ‘named’boxes; the macros are called after
\setnamebox the regular box macros, but including the string ‘name’.
\unhnamebox 1545 \providecommand*{\newnamebox}[1]{%
\unvnamebox 1546   \expandafter\newbox\csname #1\endcsname}
\namebox

```

```

1547 \providecommand*\setnamebox}[1]{%
1548   \expandafter\setbox\csname #1\endcsname}
1549 \providecommand*\unhnamebox}[1]{%
1550   \expandafter\unhbox\csname #1\endcsname}
1551 \providecommand*\unvnamebox}[1]{%
1552   \expandafter\unvbox\csname #1\endcsname}
1553 \providecommand*\namebox}[1]{%
1554   \csname #1\endcsname}
1555

```

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

```

\usenamecount 1556 \providecommand*\newnamecount}[1]{%
1557   \expandafter\newcount\csname #1\endcsname}
1558 \providecommand*\usenamecount}[1]{%
1559   \csname #1\endcsname}
1560

```

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.

```

1561 \newcount\l@dc@maxchunks
1562 \newcommand*\maxchunks}[1]{\l@dc@maxchunks=#1}
1563   \maxchunks{5120}
1564

```

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

```

\l@dnumpstartsR 1565 \newcount\l@dnumpstartsR
1566

```

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

```

\l@pscR 1567 \newcount\l@dpscL
1568 \newcount\l@dpscR
1569

```

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

```

1570 \newcommand*\l@dsetuprawboxes){%
1571   \@l@dttempcntb=\l@dc@maxchunks
1572   \loop\ifnum\@l@dttempcntb>\z@
1573     \newnamebox{\l@dLcolrawbox\the\@l@dttempcntb}
1574     \newnamebox{\l@dRcolrawbox\the\@l@dttempcntb}
1575     \advance\@l@dttempcntb \m@ne

```



```

1576 \repeat}
1577

```

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

```

1578 \newcommand*{\l@dssetupmaxlinecounts}{%
1579   \@l@dttempcntb=\l@dc@maxchunks
1580   \loop\ifnum\@l@dttempcntb>\z@
1581     \newnamecount{\l@dmaxlinesinpar\the\@l@dttempcntb}
1582     \advance\@l@dttempcntb \m@ne
1583   \repeat}
1584 \newcommand*{\l@dzeromaxlinecounts}{%
1585   \begingroup
1586   \@l@dttempcntb=\l@dc@maxchunks
1587   \loop\ifnum\@l@dttempcntb>\z@
1588     \global\usenamecount{\l@dmaxlinesinpar\the\@l@dttempcntb}=\z@
1589     \advance\@l@dttempcntb \m@ne
1590   \repeat
1591   \endgroup}
1592

```

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

```

1593 \AtBeginDocument{%
1594   \l@dssetuprawboxes
1595   \l@dssetupmaxlinecounts
1596   \l@dzeromaxlinecounts
1597   \l@dnumpstartsL=\z@
1598   \l@dnumpstartsR=\z@
1599   \l@dpsscL=\z@
1600   \l@dpsscR=\z@}
1601

```

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 1602 \newif\ifl@dusedbabel
\l@dusedbabeltrue 1603 \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 1604 \newif\ifl@dsamelang
1605 \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.
1606 \newcommand*{\l@dchecklang}{%
1607 \l@dsamelangfalse
1608 \edef\@tempa{\theledlanguageL}\edef\@tempb{\theledlanguageR}%
1609 \ifx\@tempa\@tempb
1610 \l@dsamelangtrue
1611 \fi}
1612

\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 \.
1613 \newcommand*{\l@dbbl@set@language}[1]{%
1614 \edef\language{#1}%
1615 \select@language{\language}%
1616 \if@filesw
1617 \protected@write\@auxout{}\string\select@language{\language}%
1618 \addtocontents{toc}{\string\select@language{\language}%
1619 \addtocontents{lof}{\string\select@language{\language}%
1620 \addtocontents{lot}{\string\select@language{\language}%
1621 \fi}
1622

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 1623 \providecommand{\selectlanguage}[1]{}
1624 \newcommand*{\l@duselanguage}[1]{}
1625 \gdef\theledlanguageL{}
1626 \gdef\theledlanguageR{}
1627

```

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

```
1628 \AtBeginDocument{%
1629   \ifundefined{xpg@main@language}{%
1630     \ifundefined{bbl@main@language}{%
```

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

```
1631   \l@dusedbabelfalse
1632   \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.

```
1633   \l@dusedbabeltrue
1634   \let\l@doldselectlanguage\selectlanguage
1635   \let\l@doldbbl@set@language\bbl@set@language
1636   \let\bbl@set@language\l@dbbl@set@language
1637   \renewcommand{\selectlanguage}[1]{%
1638     \l@doldselectlanguage{#1}%
1639     \ifledRcol \gdef\theledlanguageR{#1}%
1640     \else      \gdef\theledlanguageL{#1}%
1641     \fi}
```

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

```
1642   \renewcommand*{\l@duselanguage}[1]{%
1643     \l@doldselectlanguage{#1}}
```

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

```
1644   \gdef\theledlanguageL{\bbl@main@language}%
1645   \gdef\theledlanguageR{\bbl@main@language}%
1646   }%
1647 }
```

If on `Polyglossia`

```
1648 { \apptocmd{\xpg@set@language}{%
1649   \ifledRcol \gdef\theledlanguageR{#1}%
1650   \else      \gdef\theledlanguageL{#1}%
1651   \fi}%
1652   \let\l@duselanguage\xpg@set@language
1653   \gdef\theledlanguageL{\xpg@main@language}%
1654   \gdef\theledlanguageR{\xpg@main@language}%
1655 % \end{macrocode}
1656 % That's it.
1657 % \begin{macrocode}
1658 }}
```

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.

```

1659 \newcommand*{\Columns}{%
1660   \setcounter{pstartL}{\value{pstartLold}}
1661   \setcounter{pstartR}{\value{pstartRold}}
1662   \ifnum\l@dnumpstartsL=\l@dnumpstartsR\else
1663     \led@err@BadLeftRightPstarts{\the\l@dnumpstartsL}{\the\l@dnumpstartsR}%
1664   \fi

```

Start a group and zero counters, etc.

```

1665   \begingroup
1666     \l@dzero penalties
1667     \endgraf\global\num@lines=\prevgraf
1668     \global\num@linesR=\prevgraf
1669     \global\par@line=\z@
1670     \global\par@lineR=\z@
1671     \global\l@dpscL=\z@
1672     \global\l@dpscR=\z@

```

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

```

1673   \check@pstarts
1674   \loop\if@pstarts
1675     \global\pstartnumtrue
1676     \global\pstartnumRtrue

```

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

```

1677   \global\advance\l@dpscL \@ne
1678   \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

```

1679   \checkraw@text
1680   \l@dchecklang
1681 {   \loop\ifaraw@text

```

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

```

1682     \ifl@dsamelang
1683       \do@lineL
1684       \do@lineR
1685     \else
1686       \l@duselanguage{\theledlanguageL}%
1687       \do@lineL
1688       \l@duselanguage{\theledlanguageR}%
1689       \do@lineR
1690     \fi
1691     \hb@xt@ \hsize{%
1692       \hfill \unhbox\l@dleftbox
1693       \hfill \columnseparator \hfill
1694       \unhbox\l@drightbox

```

```

1695      }%
1696      \checkraw@text
1697      \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.

```

1698      \@writelinesinparL
1699      \@writelinesinparR
1700      \check@pstarts
1701      \ifbypstart@
1702          \write\linenum@out{\string\@set[1]}
1703          \resetprevline@
1704      \fi
1705      \ifbypstart@R
1706          \write\linenum@outR{\string\@set[1]}
1707          \resetprevline@
1708      \fi
1709      \addtocounter{pstartL}{1}
1710      \addtocounter{pstartR}{1}
1711      \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.

```

1712      \flush@notes
1713      \flush@notesR
1714      \endgroup
1715      \global\l@dpscL=\z@
1716      \global\l@dpscR=\z@
1717      \global\l@dnumpstartsL=\z@
1718      \global\l@dnumpstartsR=\z@
1719      \ignorespaces
1720      \global\instanzaLfalse
1721      \global\instanzaRfalse}
1722

```

`\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).

```

1723 \newcommand*{\columnseparator}{%
1724   \smash{\rule[-0.2\baselineskip]{\columnrulewidth}{1.05\baselineskip}}}
1725 \newdimen\columnrulewidth
1726   \columnrulewidth=\z@
1727

```

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

```

\@pstartstrue 1728 \newif\if@pstarts
\@pstartsfalse 1729 \newcommand*{\check@pstarts}{%
\check@pstarts

```

```

1730 \@pstartsfalse
1731 \ifnum\l@dnumpstartsL>\l@dpscL
1732   \@pstartstrue
1733 \else
1734   \ifnum\l@dnumpstartsR>\l@dpscR
1735     \@pstartstrue
1736   \fi
1737 \fi
1738 }
1739

\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
1740 \newif\ifaraw@text
1741 \araw@textfalse
1742 \newcommand*{\checkraw@text}{%
1743   \araw@textfalse
1744   \ifvbox\namebox{1@dLcolrawbox\the\l@dpscL}
1745     \araw@texttrue
1746   \else
1747     \ifvbox\namebox{1@dRcolrawbox\the\l@dpscR}
1748       \araw@texttrue
1749     \fi
1750   \fi
1751 }
1752

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

1753 \newcommand*{\@writelinesinparL}{%
1754   \edef\next{%
1755     \write\linenum@out{\string\@pend[\the\@donereallinesL]}}%
1756   \next
1757   \global\@donereallinesL \z@}
1758 \newcommand*{\@writelinesinparR}{%
1759   \edef\next{%
1760     \write\linenum@outR{\string\@pendR[\the\@donereallinesR]}}%
1761   \next
1762   \global\@donereallinesR \z@}
1763

```

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`

```

1764 \newcount\numpagelinesL
1765 \newcount\numpagelinesR
1766 \newcount\l@dminpagelines
1767

```

\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.

```

1768 \newcommand*{\Pages}{%
1769   \setcounter{pstartL}{\value{pstartLold}}
1770   \setcounter{pstartR}{\value{pstartRold}}
1771   \typeout{}
1772   \typeout{***** PAGES *****}
1773   \ifnum\l@dnumpstartsL=\l@dnumpstartsR\else
1774     \led@err@BadLeftRightPstarts{\the\l@dnumpstartsL}{\the\l@dnumpstartsR}%
1775   \fi

```

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

```

1776 \cleartol@evenpage
1777 \begingroup
1778   \l@dzeropenalties
1779   \endgraf\global\num@lines=\prevgraf
1780   \global\num@linesR=\prevgraf
1781   \global\par@line=\z@
1782   \global\par@lineR=\z@
1783   \global\l@dpscL=\z@
1784   \global\l@dpscR=\z@
1785   \writtenlinesLfalse
1786   \writtenlinesRfalse

```

Check if there are chunks to be processed.

```

1787   \check@pstarts
1788   \loop\if@pstarts

```

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

```

1789     \global\advance\l@dpscL \@ne
1790     \global\advance\l@dpscR \@ne

```

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

```

1791     \getlinesfromparlistL
1792     \getlinesfromparlistR
1793     \l@dcalc@maxoftwo{\@cs@linesinparL}{\@cs@linesinparR}%
1794     {\usernamecount{\l@dmaxlinesinpar\the\l@dpscL}}%
1795     \check@pstarts
1796   \repeat

```

Zero the counts again, ready for the next bit.

```

1797   \global\l@dpscL=\z@
1798   \global\l@dpscR=\z@

```

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

```
1799 \getlinesfrompagelistL
1800 \getlinesfrompagelistR
1801 \l@dcalc@minoftwo{\cs@linesonpageL}{\cs@linesonpageR}%
1802 {\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.

```
1803 \check@pstarts
1804 \if@pstarts
```

Increment the chunk counts to get the first pair.

```
1805 \global\advance\l@dpscL \@ne
1806 \global\advance\l@dpscR \@ne
```

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

```
1807 \global\@donereallinesL=\z@
1808 \global\@donetotallinesL=\z@
1809 \global\@donereallinesR=\z@
1810 \global\@donetotallinesR=\z@
```

Start a loop over the boxes (chunks).

```
1811 \checkraw@text
1812 % \begingroup
1813 { \loop\ifaraw@text
```

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

```
1814 \checkpageL
1815 \l@duselanguage{\theledlanguageL}%
1816 %%% \begingroup
1817 { \loop\ifl@dsamepage
1818
```

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

```
1819 \do@lineL
1820 \advance\l@mpagelinesL \@ne
1821 \ifshiftedpstarts
1822 \ifdim\ht\l@dleftbox>0pt\hb@xt@ \hsize{\ledstrutL\unhbox\l@dleftbox}
1823 \else
1824 \hb@xt@ \hsize{\ledstrutL\unhbox\l@dleftbox}%
1825 \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.

```
1826
1827 \get@nextboxL
1828 \checkpageL
1829 \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.

```

1830      \ifl@dpagfull
1831      \@writelinesonpageL{\the\numpagelinesL}%
1832      \else
1833      \@writelinesonpageL{1000}%
1834      \fi

```

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

```

1835      \numpagelinesL \z@
1836      \clearl@dleftpage }%

```

Now do the same for the right text.

```

1837      \checkpageR
1838      \l@duselanguage{\theledlanguageR}%
1839 {
1840      \loop\ifl@dsamepage
1841      \do@lineR
1842      \advance\numpagelinesR \@ne
1843      \ifshiftedpstarts
1844          \ifdim\ht\l@drightbox>0pt\hb@xt@ \hsize{\ledstrutR\unhbox\l@drightbox}\fi%
1845      \else
1846          \hb@xt@ \hsize{\ledstrutR\unhbox\l@drightbox}%
1847      \fi
1848      \get@nextboxR
1849      \checkpageR
1850      \repeat
1851      \ifl@dpagfull
1852      \@writelinesonpageR{\the\numpagelinesR}%
1853      \else
1854      \@writelinesonpageR{1000}%
1855      \fi
1856      \numpagelinesR=\z@

```

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

```

1856      \clearl@drightpage}

```

More to do? If there is we have to get the number of lines for the next pair of pages before starting to output them.

```

1857      \checkraw@text
1858      \ifaraw@text
1859      \getlinesfrompagelistL
1860      \getlinesfrompagelistR
1861      \l@dcalc@minoftwo{\@cs@linesonpageL}{\@cs@linesonpageR}%
1862      {\l@dminpagelines}%
1863      \fi
1864      \repeat}

```

We have now output the text from all the chunks.

```
1865 \fi
```

Make sure that there are no inserts hanging around.

```
1866 \flush@notes
```

```
1867 \flush@notesR
```

```
1868 \endgroup
```

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

```
1869 \global\l@dpscL=\z@
```

```
1870 \global\l@dpscR=\z@
```

```
1871 \global\l@dnumstartsL=\z@
```

```
1872 \global\l@dnumstartsR=\z@
```

```
1873 \global\instanzaLfalse
```

```
1874 \global\instanzaRfalse
```

```
1875 \ignorespaces}
```

```
1876
```

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

```
\ledstrutR 1877 \newcommand*{\ledstrutL}{\strut}
```

```
1878 \newcommand*{\ledstrutR}{\strut}
```

```
1879
```

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

```
1880 \providecommand{\cleartoevenpage}[1][\@empty]{%
```

```
1881 \clearpage
```

```
1882 \ifodd\c@page\hbox{#1}\clearpage\fi}
```

```
1883 \newcommand*{\cleartol@devenpage}{%
```

```
1884 \ifdim\pagetotal<\topskip% on an empty page
```

```
1885 \else
```

```
1886 \clearpage
```

```
1887 \fi
```

```
1888 \ifodd\c@page\hbox{#1}\clearpage\fi}
```

```
1889 \newcommand*{\clearl@dleftpage}{%
```

```
1890 \clearpage
```

```
1891 \ifodd\c@page\else
```

```
1892 \led@err@LeftOnRightPage
```

```
1893 \hbox{#1}%
```

```
1894 \cleardoublepage
```

```
1895 \fi}
```

```
1896 \newcommand*{\clearl@drightpage}{%
```

```
1897 \clearpage
```

```
1898 \ifodd\c@page
```

```
1899 \led@err@RightOnLeftPage
```

```
1900 \hbox{#1}%
```

```

1901 \cleartoevenpage
1902 \fi}
1903

```

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

```

\cs@linesinparR 1904 \newcommand*{\getlinesfromparlistL}{%
1905 \ifx\linesinpar@listL\empty
1906 \gdef\cs@linesinparL{0}%
1907 \else
1908 \glp\linesinpar@listL\to\cs@linesinparL
1909 \fi}
1910 \newcommand*{\getlinesfromparlistR}{%
1911 \ifx\linesinpar@listR\empty
1912 \gdef\cs@linesinparR{0}%
1913 \else
1914 \glp\linesinpar@listR\to\cs@linesinparR
1915 \fi}
1916

```

`\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 1917 \newcommand*{\getlinesfrompagelistL}{%
1918 \ifx\linesonpage@listL\empty
1919 \gdef\cs@linesonpageL{1000}%
1920 \else
1921 \glp\linesonpage@listL\to\cs@linesonpageL
1922 \fi}
1923 \newcommand*{\getlinesfrompagelistR}{%
1924 \ifx\linesonpage@listR\empty
1925 \gdef\cs@linesonpageR{1000}%
1926 \else
1927 \glp\linesonpage@listR\to\cs@linesonpageR
1928 \fi}
1929

```

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

```

1930 \newcommand*{\@writelinesonpageL}[1]{%
1931 \edef\next{\write\linenumout{\string\@lopL{#1}}}%
1932 \next}
1933 \newcommand*{\@writelinesonpageR}[1]{%
1934 \edef\next{\write\linenumoutR{\string\@lopR{#1}}}%
1935 \next}
1936

```

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

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

```

1937 \newcommand*{\l@dcalcmينوtwo}[3]{%
1938   \ifnum #2>#1\relax
1939     #3=#2\relax
1940   \else
1941     #3=#1\relax
1942   \fi}
1943 \newcommand*{\l@dcalcmينوtwo}[3]{%
1944   \ifnum #2<#1\relax
1945     #3=#2\relax
1946   \else
1947     #3=#1\relax
1948   \fi}
1949

```

`\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@dpagetrue` the maximum number of lines have been output then both `\ifl@dpagetrue` and
`\l@dpagetrue` `\ifl@dsamepage` are set FALSE.

```

\checkpageL 1950 \newif\ifl@dsamepage
\checkpageR 1951   \l@dsamepagetrue
1952   \newif\ifl@dpagetrue
1953   \newcommand*{\checkpageL}{%
1954     \l@dpagetrue
1955     \l@dsamepagetrue
1956     \check@goal
1957     \ifdim\pagetotal<\ledthegoal
1958       \ifnum\numpagelinesL<\l@dmينpagelines
1959       \else
1960         \l@dsamepagefalse
1961         \l@dpagetruefalse
1962       \fi
1963     \else
1964       \l@dsamepagefalse
1965       \l@dpagetrue
1966     \fi}
1967 \newcommand*{\checkpageR}{%
1968   \l@dpagetrue
1969   \l@dsamepagetrue
1970   \check@goal
1971   \ifdim\pagetotal<\ledthegoal
1972     \ifnum\numpagelinesR<\l@dmينpagelines
1973     \else
1974       \l@dsamepagefalse
1975       \l@dpagetruefalse
1976     \fi

```

```

1977 \else
1978     \l@dpagefullfalse
1979     \l@dpagefulltrue
1980 \fi}
1981

```

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

```

1982 \newdimen\ledthegoal
1983 \ifshiftedpstarts
1984     \newcommand*{\goalfraction}{0.95}
1985 \else
1986     \newcommand*{\goalfraction}{0.9}
1987 \fi
1988
1989 \newcommand*{\check@goal}{%
1990     \ledthegoal=\goalfraction\pagegoal}
1991

```

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

```

\ifwrittenlinesL 1992 \newif\ifwrittenlinesL
                  1993 \newif\ifwrittenlinesR
                  1994

```

`\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.

```

1995 \newcommand*{\get@nextboxL}{%
1996     \ifvbox\namebox{\l@dLcolrawbox\the\l@dpscl}% box is not empty

    The current box is not empty; do nothing.

1997 \else%                                box is empty

    The box is empty; check if enough lines (real and blank) have been output.
1998     \ifnum\usenamecount{\l@dmaxlinesinpar\the\l@dpscl}>\@donetotallinesL
1999     \else

        Sufficient lines have been output.

2000         \ifwrittenlinesL
2001         \else

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

2002             \@writelinesinparL
2003             \writtenlinesLtrue
2004         \fi
2005         \ifnum\l@dnumpstartsL>\l@dpscl

```

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

```

2006      \writtenlinesLfalse
2007      \ifbypstart@
2008      \ifnum\value{pstartL}<\value{pstartLold}
2009      \else
2010      \global\line@num=0
2011      \resetprevline@
2012      \fi
2013      \fi
2014      \addtocounter{pstartL}{1}
2015      \global\pstartnumtrue
2016      \l@dcalc@maxoftwo{\the\usernamecount{1@dmaxlinesinpar\the\l@dpscL}}%
2017      {\the\@donetotallinesL}%
2018      {\usernamecount{1@dmaxlinesinpar\the\l@dpscL}}%
2019      \global\@donetotallinesL \z@
2020      \global\advance\l@dpscL \@ne
2021      \fi
2022      \fi
2023      \fi}

2024 \newcommand*{\get@nextboxR}{%
2025   \ifvbox\namebox{1@dRcolrawbox\the\l@dpscR}% box is not empty
2026   \else% box is empty
2027     \ifnum\usernamecount{1@dmaxlinesinpar\the\l@dpscR}>\@donetotallinesR
2028     \else
2029       \ifwrittenlinesR
2030       \else
2031         \@writelinesinparR
2032         \writtenlinesRtrue
2033       \fi
2034       \ifnum\l@dnumpstartsR>\l@dpscR
2035         \writtenlinesRfalse
2036         \ifbypstart@R
2037         \ifnum\value{pstartR}<\value{pstartRold}
2038         \else
2039           \global\line@numR=0
2040           \resetprevline@
2041         \fi
2042         \fi
2043         \addtocounter{pstartR}{1}
2044         \global\pstartnumRtrue
2045         \l@dcalc@maxoftwo{\the\usernamecount{1@dmaxlinesinpar\the\l@dpscR}}%
2046         {\the\@donetotallinesR}%
2047         {\usernamecount{1@dmaxlinesinpar\the\l@dpscR}}%
2048         \global\@donetotallinesR \z@
2049         \global\advance\l@dpscR \@ne
2050       \fi
2051     \fi
2052   \fi}
2053

```

25 The End

i/code_i

References

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

Index

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

Symbols	
<code>\&</code>	1507, 1508, 1512, 1529, 1543
<code>\@M</code>	1518
<code>\@adv</code>	<u>352</u> , 621, 622
<code>\@afterindentfalse</code>	744
<code>\@arabic</code>	209, 210, 793, 796
<code>\@astanza@line</code>	1528, 1534, <u>1537</u>
<code>\@auxout</code>	1318, 1330, 1617
<code>\@chapter</code>	745
<code>\@cs@linesinparL</code>	1793, <u>1904</u>
<code>\@cs@linesinparR</code>	1793, <u>1904</u>
<code>\@cs@linesonpageL</code>	1801, 1861, <u>1917</u>
<code>\@cs@linesonpageR</code>	1801, 1861, <u>1917</u>
<code>\@currentlabel</code>	828, 861
<code>\@donereallinesL</code>	<u>905</u> , 934, 1755, 1757, 1807
<code>\@donereallinesR</code>	<u>905</u> , 971, 1760, 1762, 1809
<code>\@donetotallinesL</code>	<u>905</u> , 935, 938, 1808, 1998, 2017, 2019
<code>\@donetotallinesR</code>	<u>905</u> , 972, 975, 1810, 2027, 2046, 2048
<code>\@insertR</code>	1272–1274, 1287–1289
<code>\@l</code>	<u>275</u> , 590
<code>\@l@dttempcnta</code>	425, 427, 429, 430, 434, 436, 438, 439, 1025, 1064, 1065, 1067, 1069, 1072, 1073, 1090–1094, 1096, 1103, 1108, 1112, 1120, 1125, 1129, 1162, 1165, 1167, 1171
<code>\@l@dttempcntb</code>	168, 170, 172, 1055, 1056, 1103, 1108, 1112, 1120, 1125, 1129, 1154, 1158, 1171, 1179–1181, 1183, 1203–1205, 1207, 1224–1226, 1228, 1359, 1361, 1363, 1434–1436, 1438, 1571–1575, 1579–1582, 1586–1589
<code>\@l@reg</code>	324
<code>\@l@regR</code>	275
<code>\@lab</code>	544, 1309, 1321, <u>1345</u>
<code>\@lock</code>	921, 1006
<code>\@lockR</code>	60, 297, 299, 301, 314, 459, 475, 476, 478, 479, 507, 508, 510, 958, 989, 1031, 1033, 1034, 1036, 1117, 1134, 1136, 1138
<code>\@lopL</code>	<u>568</u> , 1931
<code>\@lopR</code>	<u>568</u> , 1934
<code>\@nobreakfalse</code>	802, 835

- \@nobreaktrue 800, 804, 833, 837
 \@oldnobreak 800, 802, 833, 835, 875, 891
 \@pend 559, 1755
 \@pendR 559, 1760
 \@pstartsfalse 1728
 \@pstartstrue 1728
 \@ref 531, 594, 598
 \@ref@reg 557
 \@schapter 745
 \@set 384, 628, 629, 1702, 1706
 \@tag 658, 681, 1448
 \@temp 1608
 \@templ@d 1425, 1426
 \@writelinesinparL . . 1698, 1753, 2002
 \@writelinesinparR . . 1699, 1753, 2031
 \@writelinesonpageL . 1831, 1833, 1930
 \@writelinesonpageR . 1851, 1853, 1930
 \@xloop 1285
- A**
- \absline@num 418, 432, 451, 997
 \absline@numR 58, 226, 277, 280, 283,
 415, 423, 444, 463, 497, 525,
 536, 980, 1017, 1018, 1055, 1271
 \actionlines@list
 267, 270, 418, 432, 451
 \actionlines@listR
 230, 245, 259, 262, 415,
 423, 444, 463, 497, 525, 1077, 1080
 \actions@list 271, 419, 439, 453, 455
 \actions@listR
 230, 246, 263, 416, 430, 446,
 448, 465, 474, 499, 506, 526, 1081
 \add@inserts 927
 \add@inserts@nextR 1260
 \add@insertsR 964, 1260
 \add@penaltiesL 933, 1281
 \add@penaltiesR 970, 1281
 \addtocontents 1618–1620
 \addtocounter
 877, 893, 1709, 1710, 2014, 2043
 \advancelabel@refs 1316, 1328
 \advanceline 620, 651
 \affixline@num 925
 \affixline@numR 962, 1087
 \affixpstart@numL 924, 1193
 \affixpstart@numR 961, 1193
 \affixside@note 928
 \affixside@noteR 965, 1412
 \appto 1418, 1419
- \apptocmd 1648
 \araw@textfalse 1740
 \araw@texttrue 1740
 astanza (environment) 9, 1510
 \AtBeginDocument 1341, 1593, 1628
- B**
- \ballast@count 1015, 1020
 \bbl@main@language 1644, 1645
 \bbl@set@language 1635, 1636
 \beginnumbering 7, 36, 751, 769, 807
 \beginnumberingR 49, 117, 769, 840
 \bfseries 793, 796
 \bypage@Rfalse 137, 152, 157
 \bypage@Rtrue 137, 147
 \bypstart@Rfalse 137, 148, 158
 \bypstart@Rtrue 137, 153
- C**
- \c@ballast 1020
 \c@chapter 100
 \c@chapterR 100
 \c@firstlinenumR 177, 1160
 \c@firstsublinenumR 181, 1155
 \c@linenumincrementR 177, 1160
 \c@page 590, 1882, 1888, 1891, 1898
 \c@pstartL 793
 \c@pstartR 796
 \c@section 101
 \c@sectionR 101
 \c@sublinenumincrementR 181, 1155
 \c@subsection 102
 \c@subsectionR 102
 \c@subsubsection 103
 \c@subsubsectionR 103
 \ch@ck@l@ckR 1087
 \ch@cksub@l@ckR 1087
 \ch@cksub@lockR 1156
 \chapter 731, 732, 740
 \chapterinpages 724, 732, 742
 \chardef 1507
 \check@goal 1956, 1970, 1982
 \check@pstarts
 1673, 1700, 1728, 1787, 1795, 1803
 \checkpageL 1814, 1828, 1950
 \checkpageR 1837, 1848, 1950
 \checkraw@text
 1679, 1696, 1740, 1811, 1857
 \cleardoublepage 1894
 \clearl@dleftpage 1836, 1880

`\clearl@drighthpage` 1856, [1880](#)
`\cleartoevenpage` [1880](#)
`\cleartol@devenpage` 1776, [1880](#)
`\closeout` 581, 585
`\columnrulewidth` 5, [1723](#)
`\Columns` 5, [1659](#)
`\columnseparator` 5, 1693, [1723](#)
`\content` 1371, 1385, 1399
`\correcthangingL` 930, [1488](#)
`\correcthangingR` 967, [1488](#)
`\countLline` [900](#), 911
`\countRline` [900](#), 948
`\critext` [656](#)
`\csexpandonce` 1374, 1378, 1388, 1392,
 1402, 1406, 1450, 1454, 1464, 1469
`\csuse` 1463, 1468

D

`\DeclareOption` 8, 9
`\def@tempb` 155
`\dimen` 607, 608, 612–614, 618
`\divide` 1092
`\do@actions` 998
`\do@actions@fixedcodeR` [1024](#)
`\do@actions@nextR` [1024](#)
`\do@actionsR` 981, [1024](#)
`\do@ballast` 999
`\do@ballastR` 982, [1015](#)
`\do@insidelineLhook` 930, [942](#)
`\do@insidelineRhook` [942](#), 967
`\do@lineL` [910](#), 1683, 1687, 1819
`\do@lineLhook` 915, [942](#)
`\do@lineR` [947](#), 1684, 1689, 1840
`\do@lineRhook` [942](#), 952
`\do@lockoff` [494](#)
`\do@lockoffL` 518
`\do@lockoffR` [494](#)
`\do@lockon` [459](#)
`\do@lockonL` 491
`\do@lockonR` [459](#)
`\dolistloop` 1423
`\dummy@ref` 540

E

`\edfont@info` 707, 710, 716, 719
`\edlabel` [1307](#)
`\edtext` [679](#)
`\eledmac@error` 21, 24, 27, 29
`\eledmac@warning` 1424

`\empty` .. 81, 84, 259, 267, 669, 692,
 705, 714, 816, 849, 1077, 1159,
 1167, 1262–1264, 1275, 1286,
 1310, 1322, 1905, 1911, 1918, 1924
`\end@lemmas` 669, 670, 692, 693
`\endashchar` 1297
`\endgraf` 871, 887, 1667, 1779
`\endline@num` 547, 553
`\endlock` [640](#), 1516, 1525, 1530
`\endnumbering` 7, 39, [74](#), 121, 770
`\endnumberingR` 52, [74](#), 106, 116, 129, 770
`\endpage@num` 546, 553
`\endstanzaextra` 1532
`\endsub` [607](#)
`\endsubline@num` 548, 554
environments:

`astanza` 9, [1510](#)
 `Leftside` 6, [749](#)
 `pages` 5, [724](#)
 `pairs` 5, [724](#)
 `Rightside` 6, [767](#)
`\extensionchars` . 47, 66, 112, 126, 134

F

`\f@x@l@cksR` [1087](#)
`\first@linenum@out@Rfalse` .. 576, 582
`\first@linenum@out@Rtrue` [576](#)
`\firstlinenum` 6, [186](#)
`\firstsublinenum` 6, [186](#)
`\fix@page` 320, [327](#)
`\flag@end`
 . 591, 665, 675, 676, 688, 698, 699
`\flag@start`
 . 591, 664, 665, 676, 687, 688, 699
`\flush@notes` 1712, 1866
`\flush@notesR` [1284](#), 1713, 1867
`\fullstop` . 222, 1294, 1296, 1298, 1300

G

`\get@linelistfile` 255
`\get@nextboxL` 1827, [1995](#)
`\get@nextboxR` 1847, [1995](#)
`\getline@numL` 920, 996
`\getline@numR` 957, [979](#)
`\getlinesfrompagelistL`
 1799, 1859, [1917](#)
`\getlinesfrompagelistR`
 1800, 1860, [1917](#)
`\getlinesfromparlistL` ... 1791, [1904](#)
`\getlinesfromparlistR` ... 1792, [1904](#)

\glp 262, 263,
270, 271, 670, 693, 709, 718,
1080, 1081, 1268, 1272, 1287,
1313, 1325, 1908, 1914, 1921, 1927
\goalfraction 6, 1982

H

\hangingsymbol 10, 1479, 1485
\hb@xt@ ... 923, 930, 937, 960, 967,
974, 1691, 1822, 1824, 1843, 1845
\hsize 826,
859, 1691, 1822, 1824, 1843, 1845

I

\if@filesw 1616
\if@firstcolumn 1173, 1197, 1218, 1428
\if@nobreak 799, 832
\if@pstarts ... 1674, 1728, 1788, 1804
\if@RTL 665, 676, 688, 699
\ifaraw@text ... 1681, 1740, 1813, 1858
\ifautopar 825, 858
\ifbypage@ 343
\ifbypage@R 137, 333, 1059
\ifbypstart@ 561, 1701, 2007
\ifbypstart@R ... 137, 565, 1705, 2036
\ifdim 608, 612, 614,
618, 1822, 1843, 1884, 1957, 1971
\iffirst@linenum@out@R 576, 580
\ifinserthangingsymbol .. 1477, 1491
\ifinserthangingsymbolR
..... 1475, 1483, 1501
\ifinstanzaL 747, 747, 1478, 1490
\ifinstanzaR 747, 748, 1484, 1500
\ifl@d@dash 1297
\ifl@d@elin 1299, 1300
\ifl@d@esl 1300
\ifl@d@pnum 1294, 1298
\ifl@d@ssub 1296
\ifl@dpagefull 1830, 1850, 1950
\ifl@dpadding 11, 1489, 1499
\ifl@dpairing 11, 78
\ifl@dsamelang 1604, 1682
\ifl@dsamepage 1817, 1839, 1950
\ifl@dskipnumber 1150
\ifl@dusedbabel 1602
\iflabelpstart 828, 861
\ifledplinenum 1295
\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, 683,
704, 1308, 1346, 1360, 1373,
1387, 1401, 1449, 1462, 1639, 1649
\ifnoteschanged@ 88
\ifnumberedpar@ ... 809, 842, 867,
883, 1372, 1386, 1400, 1447, 1461
\ifnumbering 37, 142, 805, 864
\ifnumberingR ... 50, 75, 108, 838, 880
\ifnumberline 983, 1000, 1149
\ifnumberpstart ... 825, 858, 876, 892
\ifnumequal 1416
\ifnumgreater 1424
\ifodd 1183, 1207,
1228, 1438, 1882, 1888, 1891, 1898
\ifpst@rtedL 32, 813
\ifpst@rtedR 32, 846
\ifpstartnum 1238, 1243
\ifpstartnumR 1193
\ifshiftedpstarts 5, 1821, 1842, 1983
\ifsidepstartnum 825, 858, 1195, 1216
\ifsublines@ 220,
309, 353, 386, 393, 424, 433,
445, 452, 464, 498, 552, 554,
984, 1001, 1066, 1153, 1348, 1352
\ifvbox 912, 949, 1744, 1747, 1996, 2025
\ifvmode 1315, 1327
\ifwrittenlinesL 1992, 2000
\ifwrittenlinesR 1993, 2029
\initnumbering@reg 45
\initnumbering@sectcmd 70
\initnumbering@sectcountR ... 71, 95
\insert@count 530, 598, 661,
684, 1380, 1394, 1408, 1456, 1471
\insert@countR 531, 594, 660,
683, 1376, 1390, 1404, 1452, 1466
\inserthangingsymbolfalse 921
\inserthangingsymbolL ... 930, 1475
\inserthangingsymbolR ... 967, 1475
\inserthangingsymbolRfalse 958
\inserthangingsymbolRtrue 958
\inserthangingsymboltrue 921
\insertlines@listR
... 81, 230, 244, 536, 1264, 1268
\inserts@list
815, 1379, 1393, 1407, 1455, 1470
\inserts@listR
... 848, 1259, 1262, 1272, 1286,
1287, 1375, 1389, 1403, 1451, 1465

- \instanzaLfalse 1720, 1873
 - \instanzaLtrue 758
 - \instanzaRfalse 1721, 1874
 - \instanzaRtrue 780
 - \interlinepenalty 1518
 - \itemcount@ 1414, 1416, 1421, 1424
- L**
- \l@d@nums 707, 710, 716, 719
 - \l@d@set 401, 636, 637
 - \l@dbbl@set@language 1613, 1636
 - \l@dbfnote 1446
 - \l@dc@maxchunks 821, 823,
854, 856, 1561, 1571, 1579, 1586
 - \l@dcalc@maxoftwo
..... 1793, 1937, 2016, 2045
 - \l@dcalc@minoftwo .. 1801, 1861, 1937
 - \l@dcalcnum 1087
 - \l@dchecklang 1606, 1680
 - \l@dchset@num 276, 279, 401
 - \l@dcsnote 1369
 - \l@dcsnotetext 1423, 1426
 - \l@emptyd@ta 916, 953
 - \l@dend@stuff ... 48, 67, 113, 127, 135
 - \l@dgetline@margin 167
 - \l@dgetsidenote@margin 1358
 - \l@dld@ta 926, 963,
1174, 1186, 1198, 1210, 1219, 1231
 - \l@dleftbox
.. 897, 922, 937, 1692, 1822, 1824
 - \l@dlinenumR 212
 - \l@dlsn@te 929, 966
 - \l@dlsnote 1369
 - \l@dmake@labels 1331
 - \l@dmake@labelsR 1319, 1335
 - \l@dminpagelines
.... 1764, 1802, 1862, 1958, 1972
 - \l@dnumpstartsL . 41, 820, 821, 823,
825, 1565, 1597, 1662, 1663,
1717, 1731, 1773, 1774, 1871, 2005
 - \l@dnumpstartsR . 54, 853, 854, 856,
858, 1565, 1598, 1662, 1663,
1718, 1734, 1773, 1774, 1872, 2034
 - \l@doldbbl@set@language 1635
 - \l@doldselectlanguage 1634, 1638, 1643
 - \l@dpagefullfalse 1950
 - \l@dpagefulltrue 1950
 - \l@dpagingfalse 13, 726, 739
 - \l@dpagingtrue 734
 - \l@dpairingfalse 11, 728, 738
 - \l@dpairingtrue 725, 733
 - \l@dpscL 912, 917, 1567, 1599, 1671,
1677, 1715, 1731, 1744, 1783,
1789, 1794, 1797, 1805, 1869,
1996, 1998, 2005, 2016, 2018, 2020
 - \l@dpscR 949, 954, 1568, 1600,
1672, 1678, 1716, 1734, 1747,
1784, 1790, 1798, 1806, 1870,
2025, 2027, 2034, 2045, 2047, 2049
 - \l@d@rd@ta 930, 967,
1176, 1184, 1200, 1208, 1221, 1229
 - \l@drightbox
.. 897, 959, 974, 1694, 1843, 1845
 - \l@d@rsn@te 931, 968
 - \l@d@rsnote 1369
 - \l@dsamelangfalse 1604, 1607
 - \l@dsamelangtrue 1604, 1610
 - \l@dsamepagefalse 1950
 - \l@dsamepagetrue 1950
 - \l@dsetupmaxlinecounts .. 1578, 1595
 - \l@dsetuprawboxes 1570, 1594
 - \l@dskipnumberfalse 1151
 - \l@dskipnumbertrue 1047
 - \l@dunhbox@line 930, 967
 - \l@dusedb@babelfalse 1602, 1631
 - \l@dusedbabeltrue 1602, 1633
 - \l@duselanguage
.... 1623, 1686, 1688, 1815, 1838
 - \l@dzeromaxlinecounts ... 1578, 1596
 - \l@dzeropenalties 870, 886, 1666, 1778
 - \l@pscL 1567
 - \l@pscR 1567
 - \label@crefs
1311, 1313, 1319, 1323, 1325, 1331
 - \labelref@list 1322, 1325, 1353
 - \labelref@listR 1305, 1310, 1313, 1349
 - \languagename .. 1614, 1615, 1617–1620
 - \last@page@num 341, 347
 - \last@page@numR 327
 - \lastbox 919, 956
 - \lastskip 607, 613
 - \Lcolwidth 5, 6, 15, 735, 826, 923, 937
 - \led@err@BadLeftRightPstarts ...
..... 23, 1663, 1774
 - \led@err@LeftOnRightPage ... 26, 1892
 - \led@err@LineationInNumbered ... 143
 - \led@err@NumberingNotStarted ... 92
 - \led@err@numberingShouldHaveStarted
..... 115
 - \led@err@NumberingStarted 38, 51

- \led@err@PendNoPstart 868, 884
 - \led@err@PendNotNumbered . . . 865, 881
 - \led@err@PstartInPstart . . . 810, 843
 - \led@err@PstartNotNumbered . . 806, 839
 - \led@err@RightOnLeftPage . . . 26, 1899
 - \led@err@TooManyPstarts 20, 822, 855
 - \led@mess@NotesChanged 89
 - \led@mess@SectionContinued 111, 125, 133
 - \led@warn@BadAction 1049
 - \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 1337
 - \ledllfill 930, 967
 - \ledRcolfalse 14, 750, 782
 - \ledRcoltrue 768
 - \ledrlfill 930, 967
 - \ledsavedprintlines 8, 1292
 - \ledstrutL 1822, 1824, 1877
 - \ledstrutR 1843, 1845, 1877
 - \ledthegoal 1957, 1971, 1982
 - \leftlinenumR 212, 1174, 1186
 - \leftpstartnumL 1193
 - \leftpstartnumR 1193
 - Leftside (environment) 6, 749
 - \Leftsidehook 756, 762
 - \Leftsidehookend 761, 762
 - \line@list 714, 718
 - \line@list@stuff 47, 126
 - \line@list@stuffR . . 66, 112, 134, 578
 - \line@listR . 84, 230, 243, 554, 705, 709
 - \line@margin 172, 1203
 - \line@marginR 165, 1179, 1224
 - \line@num . 344, 375, 376, 378, 396, 407, 408, 436, 561, 1007, 1351, 2010
 - \line@numR 59, 219, 226, 281, 315, 334, 369, 370, 372, 389, 403, 404, 427, 547, 551, 565, 990, 1060, 1069, 1158, 1160, 1162, 1163, 1347, 1424, 2039
 - \lineation 777
 - \lineationR 141, 777
 - \linenum@out 597, 605, 610, 616, 622, 629, 637, 642, 646, 1321, 1702, 1755, 1931
 - \linenum@outR 575, 581, 583, 585, 586, 590, 593, 603, 609, 615, 621, 628, 636, 641, 645, 650, 1309, 1706, 1760, 1934
 - \linenumberlist 1159, 1163
 - \linenumincrement 6, 186
 - \linenummargin 165
 - \linenumr@p 1295, 1299, 1347, 1351
 - \linenumrepR 209, 219
 - \linenumsep 214, 216, 1240, 1243, 1252, 1255
 - \linesinpar@listL 235, 251, 562, 1905, 1908
 - \linesinpar@listR 235, 247, 566, 1911, 1914
 - \linesonpage@listL 252, 570, 1918, 1921
 - \linesonpage@listR 248, 573, 1924, 1927
 - \list@clear 243–248, 251, 252, 254, 815, 848
 - \list@clearing@reg 250
 - \list@create 230–233, 235–237, 1259, 1305
 - \lock@disp 1119, 1123, 1128
 - \lock@off 485, 486, 494, 645, 646
 - \lock@on 641, 642
- M**
- \maxchunks 4, 1561
 - \maxlinesinpar@list 235, 254
 - \memorydump 8, 755, 773
 - \memorydumpL 120, 755
 - \memorydumpR 120, 773
 - \message 46, 65
 - \multiply 1093
- N**
- \n@num 522, 650
 - \n@num@reg 528
 - \namebox 912, 917, 949, 954, 1545, 1744, 1747, 1996, 2025
 - \NeedsTeXFormat 2
 - \new@line 930
 - \new@lineR 589, 967
 - \newbox 788, 897, 898, 1546
 - \newcounter 95–98, 177, 179, 181, 183, 791, 792, 794, 795
 - \newif . 5, 12, 33, 137, 138, 576, 747, 748, 1247, 1475, 1602, 1604, 1728, 1740, 1950, 1952, 1992, 1993
 - \newnamebox 1545, 1573, 1574

- \newnamecount 1556, 1581
 - \newwrite 575
 - \next@action 271
 - \next@actionline 268, 270
 - \next@actionlineR
 - . 260, 262, 1018, 1056, 1078, 1080
 - \next@actionR 263, 1019,
 - 1057, 1058, 1063, 1064, 1072, 1081
 - \next@insert 816
 - \next@insertR
 - 849, 1263, 1266, 1268, 1271, 1275
 - \next@page@num 348, 419
 - \next@page@numR 63, 284, 286, 338, 416
 - \no@expands 658, 681
 - \normal@pars 77, 819, 852
 - \normalbfnoteX 1460
 - \noteschanged@true
 - 82, 85, 706, 715, 1265
 - \num@lines 871, 1667, 1779
 - \num@linesR 787, 887, 1668, 1780
 - \numberedpar@true 827, 860
 - \numberingRfalse 76
 - \numberingRtrue 56, 106, 130
 - \numberingtrue 43, 122
 - \numberpstartfalse 9
 - \numberpstarttrue 9
 - \numdef 1414, 1421
 - \numlabfont 219
 - \numpagelinesL
 - 1764, 1820, 1831, 1835, 1958
 - \numpagelinesR
 - 1764, 1841, 1851, 1855, 1972
- O**
- \oldchapter 731, 740
 - \oldstanza 757, 758, 760, 779, 780, 783
 - \one@line 917, 919, 930
 - \one@lineR 787, 954, 956, 967
 - \openout 583, 586
- P**
- \p@pstartL 829
 - \p@pstartR 862
 - \page@action 285, 413, 541
 - \page@num 266, 346, 1205, 1436
 - \page@numR 239, 258, 336,
 - 546, 551, 1058, 1181, 1226, 1424
 - \pagegoal 1990
 - \Pages 5, 1768
 - pages (environment) 5, 724
 - \pagetotal 1884, 1957, 1971
 - pairs (environment) 5, 724
 - \par@line 872, 1669, 1781
 - \par@lineR 787, 888, 1670, 1782
 - \pausenumbering 771
 - \pausenumberingR 105, 771
 - \pend 6, 754, 776, 811, 1531
 - \pendL 754, 864
 - \pendR 776, 844, 880
 - \prevgraf
 - . 871, 887, 1667, 1668, 1779, 1780
 - \printlines 1303
 - \printlinesR 8, 1292
 - \ProcessOptions 10
 - \protected@csxdef 1463, 1468
 - \protected@edef 828, 861
 - \protected@write ... 1318, 1330, 1617
 - \ProvidesPackage 3
 - \pst@rtedLfalse 32, 42
 - \pst@rtedLtrue 123, 817
 - \pst@rtedRfalse 34, 55, 79
 - \pst@rtedRtrue 109, 131, 850
 - \pstart 6, 21, 25, 752, 775, 1533
 - \pstartL 752, 790
 - \pstartnumfalse 1240, 1245
 - \pstartnumRfalse 1252, 1257
 - \pstartnumRtrue 1248, 1676, 2044
 - \pstartnumtrue 1675, 2015
 - \pstartR 775, 790
- R**
- \Rcolwidth 5, 6, 15, 736, 859, 960, 974
 - \read@linelist 241, 579
 - \rem@inder 1163, 1165–1167
 - \resetprevline@ 1703, 1707, 2011, 2040
 - \resumenumbering 772
 - \resumenumberingR 105, 772
 - \rightlinenumR 212, 1176, 1184
 - \rightpstartnumL 1193
 - \rightpstartnumR 1193
 - Rightside (environment) 6, 767
 - \Rightsidehook 762, 778
 - \Rightsidehookend 762, 784
 - \rlap 1176, 1184, 1200, 1208, 1221, 1229
 - \Rlineflag 8, 207, 219, 1295, 1299, 1339
 - \rule 1724
- S**
- \sc@n@list 1164, 1166
 - \secdef 745

- `\section@num` 44, 46, 47, 124–126
`\section@numR`
. 30, 57, 65, 66, 110–112, 132–134
`\select@language` . . . 1615, 1617–1620
`\selectlanguage` 1623
`\set@line` 659, 682, 703
`\set@line@action`
. 278, 382, 391, 398, 421, 543
`\setl@dlp@rbox` 1429, 1441
`\setl@drp@rbox` 1431, 1439
`\setline` 624
`\setlinenum` 632
`\setnamebox` 825, 858, 1545
`\setprintlines` 1293
`\shiftedpstartsfalse` 7
`\shiftedpstartstrue` 6, 8, 9
`\shiftedversesfalse` 7
`\shiftedversestrue` 6
`\showlemma` 668, 691
`\sidenote@margin` 1363, 1367
`\sidenote@marginR` 1356, 1434
`\sidenotecontent@`
1413, 1418, 1419, 1429, 1431, 1441
`\sidenotecontent@t` 1439
`\sidenotemargin` 1356
`\sidenotesep` 1419
`\skip@lockoff` 486, 494
`\skipnumbering` 9, 649
`\skipnumbering@reg` 653
`\smash` 1724
`\splittopskip` 914, 951
`\stanza` . . . 757, 758, 760, 779, 780, 783
`\stanza@count` 1513, 1527, 1538
`\stanza@hang` 1515, 1540
`\stanzaindentbase` . . 1493, 1503, 1538
`\startlock` 640
`\startstanzahook` 1511
`\startsub` 607
`\sub@action` 294, 442, 542
`\sub@change` 64, 288, 289, 295
`\sub@lock` 1002
`\sub@lockR` 61, 303, 305, 307,
310, 460, 466, 467, 469, 470,
500, 501, 503, 985, 1039, 1041,
1042, 1044, 1100, 1140, 1142, 1144
`\sub@off` 615, 616
`\sub@on` 609, 610
`\subline@num` 221, 344, 361,
362, 364, 394, 434, 1003, 1008, 1352
`\subline@numR` 222,
226, 311, 315, 334, 355, 356,
358, 387, 425, 548, 552, 986,
991, 1060, 1067, 1154, 1155, 1348
`\sublinenumincrement` 6, 186
`\sublinenumr@p` . 1296, 1300, 1348, 1352
`\sublinenumrepR` 209, 222
`\sublines@false` 62, 292, 1029
`\sublines@true` 290, 1027
`\sublock@disp` 1102, 1106, 1111
`\symplinenum` 1295
`\sza@penalty` 1522, 1526
- T**
- `\textwidth` 16, 18, 735, 736
`\theledlanguageL` 1608, 1623, 1686, 1815
`\theledlanguageR` 1608, 1623, 1688, 1838
`\thepage` 590, 1319, 1331
`\thepstart` 753, 774
`\thepstartL`
. 9, 753, 793, 825, 829, 1239, 1244
`\thepstartR`
. 9, 774, 796, 858, 862, 1251, 1256
`\thr@@` . . 469, 478, 501, 508, 1034, 1042
`\topskip` 1884
- U**
- `\unhbox` 1550,
1692, 1694, 1822, 1824, 1843, 1845
`\unhnamebox` 1545
`\unvbox` 919, 956, 1552
`\unvnamebox` 1545
`\usenamecount`
. 1514, 1521, 1556, 1588, 1794,
1998, 2016, 2018, 2027, 2045, 2047
- V**
- `\value` 814, 847,
1660, 1661, 1769, 1770, 2008, 2037
`\vbadness` 913, 950
`\vbfnoteX` 1464, 1469
`\vbox` 825, 858
`\vl@dbfnote` 1450, 1454
`\vl@dcsnote` 1402, 1406
`\vl@dlsnote` 1374, 1378
`\vl@drsnote` 1388, 1392
`\vsplit` 917, 954
- W**
- `\wd` 930, 967

<code>\writtenlinesLfalse</code>	1785, 2006	415, 416, 418, 419, 423,
<code>\writtenlinesLtrue</code>	2003		430, 432, 439, 444, 446, 448,
<code>\writtenlinesRfalse</code>	1786, 2035		451, 453, 455, 463, 465, 474,
<code>\writtenlinesRtrue</code>	2032		497, 499, 506, 525, 526, 536,
			550, 562, 566, 570, 573, 1347,
			1351, 1374, 1378, 1388, 1392,
			1402, 1406, 1450, 1454, 1464, 1469
X			
<code>\x@lemma</code>	670–672, 693–695		
<code>\xpg@main@language</code>	1653, 1654		
<code>\xpg@set@language</code>	1648, 1652		
<code>\xright@appenditem</code>			
		Z	
		<code>\zz@@@</code>	1311, 1323

Change History

v0.1		Pstart number can be printed in side	69
General: First public release	1		
v0.10		v0.12	
General: <code>\edlabel</code> commands on the right side are now correctly indicated.	1	General: New new management of hangingsymbol insertion, preventing undesirable insertions.	53
<code>\edlabel</code> commands which start a paragraph are now put in the right place.	1	v0.2	
v0.11		General: Added section of babel related code	57
General: Change <code>\do@lineL</code> and <code>\do@lineR</code> to allow line numbering by pstart (like in eledmac 0.15).	39	Fix babel problems	1
Lineation can be by pstart (like in eledmac 0.15).	16	<code>\Columns</code> : Added <code>\l@dchecklang</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	54	v0.3	
<code>\affixline@numR</code> : Changed <code>\affixline@numR</code> to allow to disable line numbering (like in eledmac 0.15).	43	General: Added <code>\do@lineLhook</code> and <code>\do@lineRhook</code>	40
<code>\Columns</code> : Line numbering by pstart.	61	Reorganize for ledarab	1
<code>\get@nextboxR</code> : Change <code>\get@nextboxL</code> and <code>\get@nextboxR</code> to allow to disable line numbering (like in eledmac 0.15).	69	<code>\affixline@numR</code> : Changed <code>\affixline@numR</code> to match new eledmac	43
		<code>\do@actions@nextR</code> : Used <code>\do@actions@fixedcode</code> in <code>\do@actionsR</code>	42
		<code>\do@lineL</code> : Added <code>\do@lineLhook</code> to <code>\do@lineL</code>	39
		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>	40

Leftside: Added hooks into Left-side environment	34	v0.4	General: No more ledparpatch. All patches are now in the main file.	1
\flag@end: Removed extraneous spaces from \flag@end	29	v0.5	General: Corrections about \section and other titles in numbered sections	1
\ifledRcol: Moved \ifl@dpairing toeledmac	13	v0.6	General: Be able to use \chapter in parallel pages.	1
\ifpst@rtedR: Moved \ifpst@rtedL toeledmac	14	v0.7	General: Option ‘shiftedverses’ which make there is no blank between two parallel verses with unequal length.	1
\l@dlinenumR: Simplified \leftlinenumR and \rightlinenumR by introducing \l@dlinenumR	19	v0.8	General: Possibility to have a symbol on each hanging of verses, like in the french typography. Redefine the commande \hangingsymbol to define the character.	1
\l@dnumpstartsR: Moved \l@dnumpstartsL toeledmac	56	v0.9	General: Possibility to number \pstart.	9
\ledsavedprintlines: Simplified \printlinesR by using \setprintlines	49		Possibility to number the pstart with the commands \numberpstarttrue.	1
\ledstrutR: Added \ledstrutL and \ledstrutR	66		\ifledRcol: Moved \iflledRcol and \ifnumberingR toeledmac	13
\normalbfnoteX: Removed extraneous spaces from \normalbfnoteX	53	v0.9.1	General: The numbering of the pstarts restarts on each \beginnumbering.	1
\Pages: Added \ledstrutL to \Pages	64	v0.9.2	General: Debug : with \Columns, the hanging indentation now runs on the left columns and the hanging symbol is shown only when \stanza is used.	1
Added \ledstrutR to \Pages	65	v0.9.3	General: \thepstartL and \thepstartR use now \bfseries and not \bf, which is deprecated and makes conflicts with memoir class.	1
\Rightsidehookend: Added \Leftsidehook, \Leftsidehookend, \Rightsidehook and \Rightsidehookend	34	v1.0	General: Compatibility witheledmac. Change name toeledpar.	1
\sublinenumrepR: Added \linenumrepR and \sublinenumrepR	19			
v0.3.a				
General: Minor \linenummargin fix	1			
v0.3.b				
General: Improved parallel page balancing	1			
v0.3.c				
General: Compatibility with Polyglossia	1			
v0.3a				
\line@marginR: Don’t just set \line@marginR in \linenummargin	17			
v0.3b				
\Pages: Added \l@dminpagelines calculation for succeeding page pairs	65			

	Debug in lineation by pstart . . . 16	v1.3.2	
v1.0.1	General: Correction on <code>\numberonlyfirstinline</code>		General: Debug with some classes. . . . 1
	with lineation by pstart or by	v1.3.3	
	page. 1		<code>\l@dbfnote</code> : Spurious space with
v1.1			footnote in right column. 53
	General: Shiftedverses becomes		<code>\l@dcnote</code> : Debug on the left
	shiftedpstarts. 1		notes of the right column. 51
v1.1.2		v1.3.4	
	<code>\affixside@noteR</code> : Remove spuri-		<code>\l@dcnote</code> : Allow to use com-
	ous space between line number		mands in sidenotes, like it was
	and line content 52		introduced by eledmac 1.0. . . . 51
v1.2		v1.3.5	
	General: Support for <code>\led<section></code>		<code>\normalbfnoteX</code> : Allows one to
	commands in parallel texts. . . . 1		redefine <code>\thefootnoteX</code> with
v1.2.1			alph when some packages are
	<code>\initnumbering@sectcountR</code> : For		loaded. 53
	the right section, the counter is		
	defined only once. 15	v1.4	
v1.3			General: Added <code>\do@insidelineLhook</code>
	General: Manage RTL language. . . 31		and <code>\do@insidelineRhook</code> . . . 40