

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

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

Abstract

The **eledmac** package, which is based on the `PLAIN TEX` 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 v1.3.5, last revised 2013/07/10.

[†]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_elat_ex*. 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}[2013/07/10 v1.3.5 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 \LaTeX counter in `\numberingR`.

```

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

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

```

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

```

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

```

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

```

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

```

11 Line counting

11.1 Choosing the system of lineation

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

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

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

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

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

`\lineationR` `\lineationR{<word>}` is the macro used to select the lineation system for right texts. Its argument is a string: either `page`, `pstart` or `section`.

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

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

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

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

```

173   \fi
174   \fi}}

```

By default put right text numbers at the right.

```

175 \line@marginR=\@ne
176

```

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

```

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

```

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

```

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

```

`\firstlinenum` These are the user's macros for changing (sub) line numbers. They are defined in `eledmac v0.7`, but just in case I have started by `\provide`ing them.

```

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

```

```

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

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

\leftlinenumR \leftlinenumR and \rightlinenumR are the macros that are called to print the
\rightlinenumR right text's marginal line numbers. Much of the code for these is common and is
\l@dlinenumR maintained in \l@dlinenumR.
212 \newcommand*{\leftlinenumR}{%
213 \l@dlinenumR
214 \kern\linenumsep}
215 \newcommand*{\rightlinenumR}{%
216 \kern\linenumsep
217 \l@dlinenumR}
218 \newcommand*{\l@dlinenumR}{%
219 \numlabfont\linenumrepR{\line@numR}\lRlineflag%
220 \ifsublines@
221 \ifnum\subline@num>\z@
222 \unskip\fullstop\sublinenumrepR{\subline@numR}%
223 \fi
224 \fi}
225

```

11.2 Line-number counters and lists

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

```

\line@numR The count \line@numR stores the line number that's used in the right text's
\subline@numR marginal line numbering and in notes. The count \subline@numR stores a sub-line
\absline@numR number that qualifies \line@numR. The count \absline@numR stores the absolute
number of lines since the start of the right text section: that is, the number we've
actually printed, no matter what numbers we attached to them.
226 \newcount\line@numR
227 \newcount\subline@numR
228 \newcount\absline@numR
229

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

```

```

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

```

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

```

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

```

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

```

239 \newcount\page@numR
240

```

11.3 Reading the line-list file

`\read@linelist` `\read@linelist{<file>}` is the control sequence that's called by `\beginnumbering` (via `\line@list@stuff`) to open and process a line-list file; its argument is the name of the file.

```

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

```

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

```

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

```

Make sure that the `\maxlinesinpar@list` is empty (otherwise things will be thrown out of kilter if there is any old stuff still hanging in there).

```

254 \list@clear{\maxlinesinpar@list}

```

Now get the file and interpret it.

```

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

```

When the reading is done, we're all through with the line-list file. All the information we needed from it will now be encoded in our list macros. Finally, we

initialize the `\next@actionline` and `\next@action` macros, which specify where and what the next action to be taken is.

```

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

```

This version of `\read@linelist` creates list macros containing data for the entire section, so they could get rather large. The `\memorydump` macro is available if you run into macro memory limitations.

11.4 Commands within the line-list file

This section defines the commands that can appear within a line-list file, except for `\@lab` which is in a later section among the cross-referencing commands it is associated with.

The macros with `action` in their names contain all the code that modifies the action-code list.

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

```

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

```

```

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

```

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

```

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

```

```

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

```

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

```

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

```

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

```

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

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

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

```

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

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

```

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

```

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


```

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

```

`\sub@action` `\sub@action` adds an entry to the action-code list to turn sub-lineation on or off, according to the current value of the `\ifsublines@` flag.

```

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

```

`\do@lockon` `\lock@on` adds an entry to the action-code list to turn line number locking on.
`\do@lockonR` The current setting of the sub-lineation flag tells us whether this applies to line numbers or sub-line numbers.

```

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

```

```

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

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

```

```

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

```

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

```

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

```

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

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

```

531   \newcount\insert@countR

```

- #2, a sequence of other line-list-file commands, executed to determine the ending line-number. (This may also include other `\@ref` commands, corresponding to uses of `\edtext` within the first argument of another instance of `\edtext`.)

The first thing `\@ref` itself does is to add the specified number of items to the `\insertlines@list` list.

```

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

```

Next, process the second argument to determine the page and line numbers for the end of this lemma. We temporarily equate `\@ref` to a different macro

that just executes its argument, so that nested `\@ref` commands are just skipped this time. Some other macros need to be temporarily redefined to suppress their action.

```

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

```

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

```

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

```

Finally, execute the second argument of `\@ref` again, to perform for real all the commands within it.

```

555   #2
556   \else

```

And when not in right text

```

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

```

`\@pend` `\@pend{<num>}` adds its argument to the `\linesinpar@listL` list, and analogously `\@pendR` for `\@pendR`. If needed, it resets line number. We start off with a `\providecommand` just in case an older version of `eledmac` is being used which does not define these macros.

```

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

```

`\@lopL` `\@lopL{<num>}` adds its argument to the `\linesonpage@listL` list, and analogously `\@lopR` for `\@lopR`. We start off with a `\providecommand` just in case an older version of `eledmac` is being used which does not define these macros.

```

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

```

```

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

```

11.5 Writing to the line-list file

We’ve now defined all the counters, lists, and commands involved in reading the line-list file at the start of a section. Now we’ll cover the commands that `eledmac` uses within the text of a section to write commands out to the line-list.

`\linenum@outR` The file for right texts will be opened on output stream `\linenum@outR`.

```

575 \newwrite\linenum@outR

```

`\iffirst@linenum@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{\inserthangingsymbolL\new@line\l@dunhbox@line{\one@line}}\correct
          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}{}
943 \newcommand*{\do@lineRhook}{}
944

\do@lineR The \do@lineR macro is called to do all the processing for a single line of right
text.
945 \newcommand*{\do@lineR}{%
946   \advance\countRline \@ne
947   \ifvbox\namebox{\l@dRcolrawbox\the\l@dpscR}%
948   {\vbadness=10000
949    \splittopskip=\z@
950    \do@lineRhook
951    \l@emptyd@ta
952    \global\setbox\one@lineR=\vsplit\namebox{\l@dRcolrawbox\the\l@dpscR}
953    to\baselineskip}%
954   \unvbox\one@lineR \global\setbox\one@lineR=\lastbox
955   \getline@numR
956   \ifnum\@lockR>\@ne\inserthangingsymbolRtrue\else\inserthangingsymbolRfalse\fi
957   \setbox\l@drightbox
958   \hb@xt@ \Rcolwidth{%
959     \affixstart@numR
960     \affixline@numR
961     \l@dld@ta
962     \add@insertsR
963     \affixside@noter
964     \l@dlsn@te
965     {\correcthanginR\ledllfill\hb@xt@ \wd\one@lineR{\inserthangingsymbolR\new@lineR\l@du
966     \l@drsn@te
967   }}%
968   \add@penaltiesR
969   \global\advance\@donereallinesR\@ne
970   \global\advance\@donetotallinesR\@ne
971 \else
972   \setbox\l@drightbox \hb@xt@ \Rcolwidth{\hspace*{\Rcolwidth}}
973   \global\advance\@donetotallinesR\@ne
974 \fi}
975
976

```

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.
977 \newcommand*{\getline@numR}{%
978   \global\advance\absline@numR \@ne

```



```

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

```

`\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.

```

1013 \newcommand*{\do@ballastR}{\global\ballast@count=\z@
1014   \begingroup
1015     \advance\absline@numR \@ne
1016     \ifnum\next@actionlineR=\absline@numR
1017       \ifnum\next@actionR>-1001
1018         \global\advance\ballast@count by -\c@ballast
1019       \fi
1020     \fi
1021   \endgroup}

```

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

`\do@actions@fixedcodeR`

`\do@actions@nextR`

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

```

1022 \newcommand*{\do@actions@fixedcodeR}{%
1023   \ifcase\@l@dtmpcnta%
1024   \or%                               % 1001
1025     \global\sublines@true
1026   \or%                               % 1002
1027     \global\sublines@false
1028   \or%                               % 1003
1029     \global\@lockR=\@ne
1030   \or%                               % 1004
1031     \ifnum\@lockR=\tw@
1032       \global\@lockR=\thr@@
1033     \else
1034       \global\@lockR=\z@
1035     \fi
1036   \or%                               % 1005
1037     \global\sub@lockR=\@ne
1038   \or%                               % 1006
1039     \ifnum\sub@lockR=\tw@
1040       \global\sub@lockR=\thr@@
1041     \else
1042       \global\sub@lockR=\z@
1043     \fi
1044   \or%                               % 1007
1045     \l@dskipnumbertrue
1046   \else
1047     \led@warn@BadAction
1048   \fi}
1049
1050
1051 \newcommand*{\do@actionsR}{%
1052   \global\let\do@actions@nextR=\relax
1053   \@l@dtmpcntb=\absline@numR
1054   \ifnum\@l@dtmpcntb<\next@actionlineR\else
1055     \ifnum\next@actionR>-1001\relax
1056       \global\page@numR=\next@actionR
1057       \ifbypage@R
1058         \global\line@numR \z@ \global\subline@numR \z@
1059       \fi
1060     \else
1061       \ifnum\next@actionR<-4999\relax    % 9/05 added relax here
1062         \@l@dtmpcnta=-\next@actionR
1063         \advance\@l@dtmpcnta by -5001\relax
1064         \ifsublines@
1065           \global\subline@numR=\@l@dtmpcnta
1066         \else
1067           \global\line@numR=\@l@dtmpcnta
1068         \fi

```

```

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

```

14.4 Line number printing

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

```

\ch@cksub@l@ckR 1085
\ch@ck@l@ckR 1086 \providecommand*{\l@dcalcnun}[3]{%
\fx@l@cksR 1087 \ifnum #1 > #2\relax
\affixline@numR 1088 \l@dttempcnta = #1\relax
1089 \advance\@l@dttempcnta by -#2\relax
1090 \divide\@l@dttempcnta by #3\relax
1091 \multiply\@l@dttempcnta by #3\relax
1092 \advance\@l@dttempcnta by #2\relax
1093 \else
1094 \l@dttempcnta=#2\relax
1095 \fi}
1096
1097 \newcommand*{\ch@cksub@l@ckR}{%
1098 \ifcase\sub@lockR
1099 \or
1100 \ifnum\sublock@disp=\@ne
1101 \l@dttempcntb \z@ \l@dttempcnta \@ne
1102 \fi
1103 \or
1104 \ifnum\sublock@disp=\tw@
1105 \else
1106 \l@dttempcntb \z@ \l@dttempcnta \@ne
1107 \fi
1108 \or
1109 \ifnum\sublock@disp=\z@
1110 \l@dttempcntb \z@ \l@dttempcnta \@ne
1111 \fi
1112 \fi}
1113

```

```

1114 \newcommand*{\ch@ck@l@ckR}{%
1115   \ifcase\@lockR
1116   \or
1117     \ifnum\lock@disp=\@ne
1118       \@l@tempcntb \z@ \@l@tempcnta \@ne
1119     \fi
1120   \or
1121     \ifnum\lock@disp=\tw@
1122     \else
1123       \@l@tempcntb \z@ \@l@tempcnta \@ne
1124     \fi
1125   \or
1126     \ifnum\lock@disp=\z@
1127       \@l@tempcntb \z@ \@l@tempcnta \@ne
1128     \fi
1129 \fi}
1130
1131 \newcommand*{\f@x@l@cksR}{%
1132   \ifcase\@lockR
1133   \or
1134     \global\@lockR \tw@
1135   \or \or
1136     \global\@lockR \z@
1137   \fi
1138   \ifcase\sub@lockR
1139   \or
1140     \global\sub@lockR \tw@
1141   \or \or
1142     \global\sub@lockR \z@
1143   \fi}
1144
1145
1146 \newcommand*{\affixline@numR}{%
1147   \ifnumberline
1148   \ifl@dskipnumber
1149     \global\l@dskipnumberfalse
1150   \else
1151     \ifsublines@
1152       \@l@tempcntb=\subline@numR
1153       \l@dcalcnm{\subline@numR}{\c@firstsublinenumR}{\c@sublinenumincrementR}%
1154       \ch@cksub@lockR
1155     \else
1156       \@l@tempcntb=\line@numR
1157       \ifx\linenumberlist\empty
1158         \l@dcalcnm{\line@numR}{\c@firstlinenumR}{\c@linenumincrementR}%
1159       \else
1160         \@l@tempcnta=\line@numR
1161         \edef\rem@inder{\linenumberlist,\number\line@numR,%}
1162         \edef\sc@n@list{\def\noexpand\sc@n@list
1163           ###1,\number\@l@tempcnta,###2|{\def\noexpand\rem@inder{###2}}}%

```

```

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

```

14.5 Pstart number printing in side

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

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

```

\affixpstart@numL
\affixpstart@numR 1191
\leftpstartnumR 1192 \newcommand*{\affixpstart@numL}{%
\rightpstartnumR 1193 \ifsidepstartnum
\leftpstartnumL 1194 \if@twocolumn
\rightpstartnumL 1195 \if@firstcolumn
\ifpstartnumR 1196 \gdef\l@dld@ta{\llap{\leftpstartnumL}}}%
1197 \else
1198 \gdef\l@drd@ta{\rlap{\rightpstartnumL}}}%
1199 \fi
1200 \else
1201 \l@tempcntb=\line@margin

```

```

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

```

```

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

```

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.

```

1257 \list@create{\inserts@listR}

```

`\add@insertsR` The right text version.

```

\add@inserts@nextR 1258 \newcommand*{\add@insertsR}{%
1259   \global\let\add@inserts@nextR=\relax
1260   \ifx\inserts@listR\empty \else
1261     \ifx\next@insertR\empty
1262       \ifx\insertlines@listR\empty
1263         \global\noteschanged@true
1264         \gdef\next@insertR{100000}%
1265       \else
1266         \gl@p\insertlines@listR\to\next@insertR
1267       \fi
1268     \fi
1269     \ifnum\next@insertR=\absline@numR
1270       \gl@p\inserts@listR\to\@insertR
1271       \@insertR
1272       \global\let\@insertR=\undefined
1273       \global\let\next@insertR=\empty
1274       \global\let\add@inserts@nextR=\add@insertsR
1275     \fi
1276   \fi
1277   \add@inserts@nextR}
1278

```

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.

```

1279 \newcommand*{\add@penaltiesL}{\}
1280 \newcommand*{\add@penaltiesR}{\}
1281

```

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.

```

1282 \newcommand*{\flush@notesR}{%
1283 \xloop
1284 \ifx\inserts@listR\empty \else
1285 \gl@p\inserts@listR\to\@insertR
1286 \@insertR
1287 \global\let\@insertR=\undefined
1288 \repeat}
1289

```


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
1290 \def\printlinesR#1|#2|#3|#4|#5|#6|#7|{\begingroup
1291   \setprintlines{#1}{#2}{#3}{#4}{#5}{#6}%
1292   \ifl@d@pnum #1\fullstop\fi
1293   \ifledplinenum \linenumr@p{#2}\Rlineflag\else \sympninenum\fi
1294   \ifl@d@ssub \fullstop \sublinenumr@p{#3}\fi
1295   \ifl@d@dash \endashchar\fi
1296   \ifl@d@pnum #4\fullstop\fi
1297   \ifl@d@elin \linenumr@p{#5}\Rlineflag\fi
1298   \ifl@d@esl \ifl@d@elin \fullstop\fi \sublinenumr@p{#6}\fi
1299 \endgroup}
1300
1301 \let\ledsavedprintlines\printlines
1302

```

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.

```

1303 \list@create{\labelref@listR}
1304

```

`\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.

```

1305 \renewcommand*{\edlabel}[1]{\@bsphack
1306   \ifledRcol
1307     \write\linenum@outR{\string\@lab}%
1308     \ifx\labelref@listR\empty
1309       \xdef\label@refs{\zz@@@}%
1310     \else

```

```

1311     \glp\labelref@listR\to\label@refs
1312     \fi
1313     \ifvmode
1314         \advancelabel@refs
1315     \fi
1316     \protected@write\@auxout{%
1317         {\string\l@dmake@labelsR\space\thepage|\label@refs|{#1}}%
1318     \else
1319         \write\linenum@out{\string\@lab}%
1320         \ifx\labelref@list\empty
1321             \xdef\label@refs{\zz@@@}%
1322         \else
1323             \glp\labelref@list\to\label@refs
1324         \fi
1325     \ifvmode
1326         \advancelabel@refs
1327     \fi
1328     \protected@write\@auxout{%
1329         {\string\l@dmake@labelsR\space\thepage|\label@refs|{#1}}%
1330     \fi
1331     \@esphack}
1332

```

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

```

1333 \def\l@dmake@labelsR#1|#2|#3|#4{%
1334     \expandafter\ifx\csname the@label#4\endcsname \relax\else
1335         \led@warn@DuplicateLabel{#4}%
1336     \fi
1337     \expandafter\gdef\csname the@label#4\endcsname{#1|#2\Rlineflag|#3}%
1338     \ignorespaces}
1339 \AtBeginDocument{%
1340     \def\l@dmake@labelsR#1|#2|#3|#4{%
1341     }
1342

```

`\@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.

```

1343 \renewcommand*{\@lab}{%
1344     \ifledRcol
1345         \xright@appenditem{\linenumr@p{\line@numR}}{|%
1346         \ifsublines@ \sublinenumr@p{\subline@numR}\else 0\fi}%
1347         \to\labelref@listR
1348     \else
1349         \xright@appenditem{\linenumr@p{\line@num}}{|%
1350         \ifsublines@ \sublinenumr@p{\subline@num}\else 0\fi}%
1351         \to\labelref@list
1352     \fi}
1353

```

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 1354 \newcount\sidenote@marginR
                  1355 \renewcommand*{\sidenotemargin}[1]{\%
                  1356 \l@dgetsidenote@margin{#1}%
                  1357 \ifnum\@l@dttempcntb>\m@ne
                  1358 \ifledRcol
                  1359 \global\sidenote@marginR=\@l@dttempcntb
                  1360 \else
                  1361 \global\sidenote@margin=\@l@dttempcntb
                  1362 \fi
                  1363 \fi}}
                  1364 \sidenotemargin{right}
                  1365 \global\sidenote@margin=\@ne
                  1366
```

`\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 1367 \renewcommand*{\l@dlsnote}[1]{\%
            1368 \begingroup%
            1369 \newcommand{\content}{#1}%
            1370 \ifnumberedpar@
            1371 \ifledRcol%
            1372 \xright@appenditem{\noexpand\l@dlsnote{\csexpandonce{content}}}%
            1373 \to\inserts@listR
            1374 \global\advance\insert@countR \@ne%
            1375 \else%
            1376 \xright@appenditem{\noexpand\l@dlsnote{\csexpandonce{content}}}%
            1377 \to\inserts@list
            1378 \global\advance\insert@count \@ne%
            1379 \fi
            1380 \fi\ignorespaces\endgroup}
            1381 \renewcommand*{\l@drsnote}[1]{\%
            1382 \begingroup%
            1383 \newcommand{\content}{#1}%
            1384 \ifnumberedpar@
            1385 \ifledRcol%
            1386 \xright@appenditem{\noexpand\l@drsnote{\csexpandonce{content}}}%
            1387 \to\inserts@listR
            1388 \global\advance\insert@countR \@ne%
            1389 \else%
            1390 \xright@appenditem{\noexpand\l@drsnote{\csexpandonce{content}}}%
            1391 \to\inserts@list
            1392 \global\advance\insert@count \@ne%
            1393 \fi
            1394 \fi\ignorespaces\endgroup}
```

```

1395 \renewcommand*{\l@dcnote}[1]{%
1396   \begingroup%
1397   \newcommand{\content}{#1}%
1398   \ifnumberedpar@
1399     \ifledRcol%
1400       \xright@appenditem{\noexpand\l@dcnote{\csexpandonce{content}}}%
1401         \to\inserts@listR
1402       \global\advance\insert@countR \@ne%
1403     \else%
1404       \xright@appenditem{\noexpand\l@dcnote{\csexpandonce{content}}}%
1405         \to\inserts@list
1406       \global\advance\insert@count \@ne%
1407     \fi
1408   \fi\ignorespaces\endgroup}
1409

```

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

```

1410 \newcommand*{\affixside@noteR}{%
1411   \def\sidenotecontent@{}%
1412   \numdef{\itemcount@}{0}%
1413   \def\do##1{%
1414     \ifnumequal{\itemcount@}{0}%
1415       {%
1416         \appto\sidenotecontent@{##1}}% Not print not separator before the 1st note
1417       {\appto\sidenotecontent@{\sidenotesep ##1}}%
1418       }%
1419     \numdef{\itemcount@}{\itemcount@+1}%
1420   }%
1421   \dolistloop{\l@dcnotetext}%
1422   \ifnumgreater{\itemcount@}{1}{\eledmac@warning{\itemcount@\space sidenotes on line \tl
1423   \gdef\@templ@d}{%
1424   \ifx\@templ@d\l@dcnotetext \else%
1425     \if@twocolumn%
1426       \if@firstcolumn%
1427         \setl@dlp@rbox{\sidenotecontent@}%
1428       \else%
1429         \setl@drp@rbox{\sidenotecontent@}%
1430       \fi%
1431     \else%
1432       \l@dttempcntb=\sidenote@marginR%
1433       \ifnum\l@dttempcntb>\@ne%
1434         \advance\l@dttempcntb by\page@num%
1435       \fi%
1436       \ifodd\l@dttempcntb%
1437         \setl@drp@rbox{\sidenotecontent@t}%
1438       \else%
1439         \setl@dlp@rbox{\sidenotecontent@}%
1440       \fi%
1441     \fi%
1442   \fi}

```

1443

18 Familiar footnotes

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

```

1444 \renewcommand{\l@dbfnote}[1]{%
1445   \ifnumberedpar@
1446   \gdef\@tag{#1}%
1447   \ifledRcol%
1448     \xright@appenditem{\noexpand\vl@dbfnote{\csexpandonce{\@tag}}{\@thefnmark}}%
1449                       \to\inserts@listR
1450     \global\advance\insert@countR \@ne%
1451   \else%
1452     \xright@appenditem{\noexpand\vl@dbfnote{\csexpandonce{\@tag}}{\@thefnmark}}%
1453                       \to\inserts@list
1454     \global\advance\insert@count \@ne%
1455   \fi
1456 \fi\ignorespaces}
1457

```

`\normalbfnoteX`

```

1458 \renewcommand{\normalbfnoteX}[2]{%
1459   \ifnumberedpar@
1460   \ifledRcol%
1461     \protected@csxdef{thisfootnote}{\csuse{thefootnote#1}}%
1462     \xright@appenditem{\noexpand\vbfnoteX{#1}{#2}{\csexpandonce{thisfootnote}}}%
1463                       \to\inserts@listR
1464     \global\advance\insert@countR \@ne%
1465   \else%
1466     \protected@csxdef{thisfootnote}{\csuse{thefootnote#1}}%
1467     \xright@appenditem{\noexpand\vbfnoteX{#1}{#2}{\csexpandonce{thisfootnote}}}%
1468                       \to\inserts@list
1469     \global\advance\insert@count \@ne%
1470   \fi
1471 \fi\ignorespaces}
1472

```

19 Verse

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

`\inserthangingsymbolL`

```

\inserthangingsymbolR 1473 \newif\ifinserthangingsymbolR
1474 \newcommand{\inserthangingsymbolL}{%
1475 \ifinserthangingsymbol%

```

```

1476     \ifistanzaL%
1477         \hfill\hangingsymbol%
1478     \fi%
1479 \fi}
1480 \newcommand{\inserthangingsymbolR}{%
1481 \ifinserthangingsymbolR%
1482     \ifistanzaR%
1483         \hfill\hangingsymbol%
1484     \fi%
1485 \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 1486 \newcommand{\correcthangingL}{%
1487 \ifl@dpaging\else%
1488     \ifistanzaL%
1489         \ifinserthangingsymbol%
1490             \hskip \@ifundefined{sza@00}{0}{\expandafter%
1491                 \noexpand\csname sza@00\endcsname}\stanzaindentbase%
1492         \fi%
1493     \fi%
1494 \fi}
1495
1496 \newcommand{\correcthangingR}{%
1497 \ifl@dpaging\else%
1498     \ifistanzaR%
1499         \ifinserthangingsymbolR%
1500             \hskip \@ifundefined{sza@00}{0}{\expandafter%
1501                 \noexpand\csname sza@00\endcsname}\stanzaindentbase%
1502         \fi%
1503     \fi%
1504 \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.

```

1505 \chardef\next=\catcode'\&
1506 \catcode'\&=\active
1507

```

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

```

1508 \newenvironment{astanza}{%
1509     \startstanzahook
1510     \catcode'\&\active

```

```

1511 \global\stanza@count\@ne
1512 \ifnum\usernamecount{sza@00}=\z@
1513   \let\stanza@hang\relax
1514   \let\endlock\relax
1515 \else
1516 %%    \interlinepenalty\@M % this screws things up, but I don't know why
1517   \rightskip\z@ plus 1fil\relax
1518 \fi
1519 \ifnum\usernamecount{szp@00}=\z@
1520   \let\sza@penalty\relax
1521 \fi
1522 \def&{%
1523   \endlock\mbox{}}%
1524   \sza@penalty
1525   \global\advance\stanza@count\@ne
1526   \@astanza@line}%
1527 \def\&{%
1528   \endlock\mbox{}}
1529   \pend
1530   \endstanzaextra}%
1531 \pstart
1532 \@astanza@line
1533 }{}
1534

```

`\@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.

```

1535 \newcommand*{\@astanza@line}{%
1536   \parindent=\csname sza@\number\stanza@count @\endcsname\stanzaindentbase
1537   \par
1538   \stanza@hang%\mbox{}}%
1539   \ignorespaces}
1540

```

Lastly reset the modified category codes.

```

1541 \catcode'\&=\next
1542

```

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 1543 \providecommand*{\newnamebox}[1]{%
\unvnamebox 1544 \expandafter\newbox\csname #1\endcsname}
\namebox

```

```

1545 \providecommand*\setnamebox}[1]{%
1546   \expandafter\setbox\csname #1\endcsname}
1547 \providecommand*\unhnamebox}[1]{%
1548   \expandafter\unhbox\csname #1\endcsname}
1549 \providecommand*\unvnamebox}[1]{%
1550   \expandafter\unvbox\csname #1\endcsname}
1551 \providecommand*\namebox}[1]{%
1552   \csname #1\endcsname}
1553

```

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

```

\usernamecount 1554 \providecommand*\newnamecount}[1]{%
1555   \expandafter\newcount\csname #1\endcsname}
1556 \providecommand*\usernamecount}[1]{%
1557   \csname #1\endcsname}
1558

```

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.

```

1559 \newcount\l@dc@maxchunks
1560 \newcommand*\maxchunks}[1]{\l@dc@maxchunks=#1}
1561 \maxchunks{5120}
1562

```

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

```

\l@dnumpstartsR 1563 \newcount\l@dnumpstartsR
1564

```

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

```

\l@pscR 1565 \newcount\l@dpscL
1566 \newcount\l@dpscR
1567

```

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

```

1568 \newcommand*\l@dsetuprawboxes){%
1569   \@l@dttempcntb=\l@dc@maxchunks
1570   \loop\ifnum\@l@dttempcntb>\z@
1571     \newnamebox{\l@dLcolrawbox\the\@l@dttempcntb}
1572     \newnamebox{\l@dRcolrawbox\the\@l@dttempcntb}
1573     \advance\@l@dttempcntb \m@ne

```



```
1574 \repeat}
1575
```

`\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.

```
1576 \newcommand*{\l@dssetupmaxlinecounts}{%
1577   \@l@dttempcntb=\l@dc@maxchunks
1578   \loop\ifnum\@l@dttempcntb>\z@
1579     \newnamecount{\l@dmaxlinesinpar\the\@l@dttempcntb}
1580     \advance\@l@dttempcntb \m@ne
1581   \repeat}
1582 \newcommand*{\l@dzeromaxlinecounts}{%
1583   \begingroup
1584   \@l@dttempcntb=\l@dc@maxchunks
1585   \loop\ifnum\@l@dttempcntb>\z@
1586     \global\usenamecount{\l@dmaxlinesinpar\the\@l@dttempcntb}=\z@
1587     \advance\@l@dttempcntb \m@ne
1588   \repeat
1589   \endgroup}
1590
```

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

```
1591 \AtBeginDocument{%
1592   \l@dssetuprawboxes
1593   \l@dssetupmaxlinecounts
1594   \l@dzeromaxlinecounts
1595   \l@dnumpstartsL=\z@
1596   \l@dnumpstartsR=\z@
1597   \l@dpsscL=\z@
1598   \l@dpsscR=\z@}
1599
```

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 1600 \newif\ifl@dusedbabel
\l@dusedbabeltrue 1601 \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 1602 \newif\ifl@dsamelang
1603 \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.
1604 \newcommand*{\l@dchecklang}{%
1605 \l@dsamelangfalse
1606 \edef\@tempa{\theledlanguageL}\edef\@tempb{\theledlanguageR}%
1607 \ifx\@tempa\@tempb
1608 \l@dsamelangtrue
1609 \fi}
1610

\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 \.
1611 \newcommand*{\l@dbbl@set@language}[1]{%
1612 \edef\languageName{#1}%
1613 \select@language{\languageName}%
1614 \if@filesw
1615 \protected@write\@auxout{}\string\select@language{\languageName}%
1616 \addtocontents{toc}{\string\select@language{\languageName}%
1617 \addtocontents{lof}{\string\select@language{\languageName}%
1618 \addtocontents{lot}{\string\select@language{\languageName}%
1619 \fi}
1620

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 1621 \providecommand{\selectlanguage}[1]{%
1622 \newcommand*{\l@duselanguage}[1]{%
1623 \gdef\theledlanguageL{
1624 \gdef\theledlanguageR{
1625

```

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

```
1626 \AtBeginDocument{%
1627   \@ifundefined{xpg@main@language}{%
1628     \@ifundefined{bbl@main@language}{%
```

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

```
1629   \l@dusedbabelfalse
1630   \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.

```
1631   \l@dusedbabeltrue
1632   \let\l@doldselectlanguage\selectlanguage
1633   \let\l@doldbbl@set@language\bbl@set@language
1634   \let\bbl@set@language\l@dbbl@set@language
1635   \renewcommand{\selectlanguage}[1]{%
1636     \l@doldselectlanguage{#1}%
1637     \ifledRcol \gdef\theledlanguageR{#1}%
1638     \else      \gdef\theledlanguageL{#1}%
1639     \fi}
```

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

```
1640   \renewcommand*{\l@duselanguage}[1]{%
1641     \l@doldselectlanguage{#1}}
```

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

```
1642   \gdef\theledlanguageL{\bbl@main@language}%
1643   \gdef\theledlanguageR{\bbl@main@language}%
1644   }%
1645 }
```

If on `Polyglossia`

```
1646 { \apptocmd{\xpg@set@language}{%
1647   \ifledRcol \gdef\theledlanguageR{#1}%
1648   \else      \gdef\theledlanguageL{#1}%
1649   \fi}%
1650   \let\l@duselanguage\xpg@set@language
1651   \gdef\theledlanguageL{\xpg@main@language}%
1652   \gdef\theledlanguageR{\xpg@main@language}%
1653 % \end{macrocode}
1654 % That's it.
1655 %   \begin{macrocode}
1656 }}
```

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.

```

1657 \newcommand*{\Columns}{%
1658   \setcounter{pstartL}{\value{pstartLold}}
1659   \setcounter{pstartR}{\value{pstartRold}}
1660   \ifnum\l@dnumstartsL=\l@dnumstartsR\else
1661     \led@err@BadLeftRightPstarts{\the\l@dnumstartsL}{\the\l@dnumstartsR}%
1662   \fi

```

Start a group and zero counters, etc.

```

1663 \begingroup
1664   \l@dzeropenalties
1665   \endgraf\global\num@lines=\prevgraf
1666   \global\num@linesR=\prevgraf
1667   \global\par@line=\z@
1668   \global\par@lineR=\z@
1669   \global\l@dpscL=\z@
1670   \global\l@dpscR=\z@

```

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

```

1671   \check@pstarts
1672   \loop\if@pstarts
1673     \global\pstartnumtrue
1674     \global\pstartnumRtrue

```

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

```

1675   \global\advance\l@dpscL \@ne
1676   \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

```

1677   \checkraw@text
1678   \l@dchecklang
1679 {   \loop\ifaraw@text

```

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

```

1680     \ifl@dsamelang
1681       \do@lineL
1682       \do@lineR
1683     \else
1684       \l@duselanguage{\theledlanguageL}%
1685       \do@lineL
1686       \l@duselanguage{\theledlanguageR}%
1687       \do@lineR
1688     \fi
1689     \hb@xt@ \hsize{%
1690       \hfill \unhbox\l@dleftbox
1691       \hfill \columnseparator \hfill
1692       \unhbox\l@drightbox

```

```

1693     }%
1694     \checkraw@text
1695     \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.

```

1696     \@writelinesinparL
1697     \@writelinesinparR
1698     \check@pstarts
1699     \ifbypstart@
1700         \write\linenum@out{\string\@set[1]}
1701         \resetprevline@
1702     \fi
1703     \ifbypstart@R
1704         \write\linenum@outR{\string\@set[1]}
1705         \resetprevline@
1706     \fi
1707     \addtocounter{pstartL}{1}
1708     \addtocounter{pstartR}{1}
1709     \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.

```

1710     \flush@notes
1711     \flush@notesR
1712     \endgroup
1713     \global\l@dpscL=\z@
1714     \global\l@dpscR=\z@
1715     \global\l@dnumpstartsL=\z@
1716     \global\l@dnumpstartsR=\z@
1717     \ignorespaces
1718     \global\instanzaLfalse
1719     \global\instanzaRfalse}
1720

```

`\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).

```

1721 \newcommand*{\columnseparator}{%
1722     \smash{\rule[-0.2\baselineskip]{\columnrulewidth}{1.05\baselineskip}}}
1723 \newdimen\columnrulewidth
1724     \columnrulewidth=\z@
1725

```

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

```

\@pstartstrue 1726 \newif\if@pstarts
\@pstartsfalse 1727 \newcommand*{\check@pstarts}{%
\check@pstarts

```

```

1728 \@pstartsfalse
1729 \ifnum\l@dnumpstartsL>\l@dpscL
1730   \@startstrue
1731 \else
1732   \ifnum\l@dnumpstartsR>\l@dpscR
1733     \@startstrue
1734   \fi
1735 \fi
1736 }
1737

\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
1738 \newif\ifaraw@text
1739 \araw@textfalse
1740 \newcommand*{\checkraw@text}{%
1741   \araw@textfalse
1742   \ifvbox\namebox{1@dLcolrawbox\the\l@dpscL}
1743     \araw@texttrue
1744   \else
1745     \ifvbox\namebox{1@dRcolrawbox\the\l@dpscR}
1746       \araw@texttrue
1747     \fi
1748   \fi
1749 }
1750

\@writelinesinparL These write the number of text lines in a chunk to the section files, and then
\@writelinesinparR afterwards zero the counter.
1751 \newcommand*{\@writelinesinparL}{%
1752   \edef\next{%
1753     \write\linenum@out{\string\@pend[\the\@donereallinesL]}}%
1754   \next
1755   \global\@donereallinesL \z@}
1756 \newcommand*{\@writelinesinparR}{%
1757   \edef\next{%
1758     \write\linenum@outR{\string\@pendR[\the\@donereallinesR]}}%
1759   \next
1760   \global\@donereallinesR \z@}
1761

```

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`

```

1762 \newcount\numpagelinesL
1763 \newcount\numpagelinesR
1764 \newcount\l@dminpagelines
1765

```

\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.

```

1766 \newcommand*{\Pages}{%
1767   \setcounter{pstartL}{\value{pstartLold}}
1768   \setcounter{pstartR}{\value{pstartRold}}
1769   \typeout{}
1770   \typeout{***** PAGES *****}
1771   \ifnum\l@dnumpstartsL=\l@dnumpstartsR\else
1772     \led@err@BadLeftRightPstarts{\the\l@dnumpstartsL}{\the\l@dnumpstartsR}%
1773   \fi

```

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

```

1774 \cleartol@evenpage
1775 \begingroup
1776   \l@dzeropenalties
1777   \endgraf\global\l@num@lines=\prevgraf
1778   \global\l@num@linesR=\prevgraf
1779   \global\l@par@line=\z@
1780   \global\l@par@lineR=\z@
1781   \global\l@dpscL=\z@
1782   \global\l@dpscR=\z@
1783   \writtenlinesLfalse
1784   \writtenlinesRfalse

```

Check if there are chunks to be processed.

```

1785   \check@pstarts
1786   \loop\if@pstarts

```

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

```

1787     \global\advance\l@dpscL \@ne
1788     \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**.

```

1789     \getlinesfromparlistL
1790     \getlinesfromparlistR
1791     \l@dcalc@maxoftwo{\cs@linesinparL}{\cs@linesinparR}%
1792     {\usenamecount{\l@dmaxlinesinpar\the\l@dpscL}}%
1793     \check@pstarts
1794   \repeat

```

Zero the counts again, ready for the next bit.

```

1795   \global\l@dpscL=\z@
1796   \global\l@dpscR=\z@

```

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

```
1797 \getlinesfrompagelistL
1798 \getlinesfrompagelistR
1799 \l@dcalc@minoftwo{\@cs@linesonpageL}{\@cs@linesonpageR}%
1800 {\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.

```
1801 \check@pstarts
1802 \if@pstarts
```

Increment the chunk counts to get the first pair.

```
1803 \global\advance\l@dpscL \@ne
1804 \global\advance\l@dpscR \@ne
```

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

```
1805 \global\@donereallinesL=\z@
1806 \global\@donetotallinesL=\z@
1807 \global\@donereallinesR=\z@
1808 \global\@donetotallinesR=\z@
```

Start a loop over the boxes (chunks).

```
1809 \checkraw@text
1810 % \begingroup
1811 { \loop\ifaraw@text
```

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

```
1812 \checkpageL
1813 \l@ducelanguage{\theledlanguageL}%
1814 %%% \begingroup
1815 { \loop\ifl@dsamepage
1816
```

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

```
1817 \do@lineL
1818 \advance\l@numpagelinesL \@ne
1819 \ifshiftedpstarts
1820 \ifdim\ht\l@dleftbox>0pt\hb@xt@ \hsize{\ledstrutL\unhbox\l@dleftbox}
1821 \else
1822 \hb@xt@ \hsize{\ledstrutL\unhbox\l@dleftbox}%
1823 \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.

```
1824
1825 \get@nextboxL
1826 \checkpageL
1827 \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.

```

1828      \ifl@dpagfull
1829      \@writelinesonpageL{\the\numpagelinesL}%
1830      \else
1831      \@writelinesonpageL{1000}%
1832      \fi

```

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

```

1833      \numpagelinesL \z@
1834      \clearl@dleftpage }%

```

Now do the same for the right text.

```

1835      \checkpageR
1836      \l@duselanguage{\theledlanguageR}%
1837 {
1838      \loop\ifl@dsamepage
1839      \do@lineR
1840      \advance\numpagelinesR \@ne
1841      \ifshiftedpstarts
1842          \ifdim\ht\l@drightbox>0pt\hb@xt@ \hsize{\ledstrutR\unhbox\l@drightbox}\fi%
1843      \else
1844          \hb@xt@ \hsize{\ledstrutR\unhbox\l@drightbox}%
1845      \fi
1846      \get@nextboxR
1847      \checkpageR
1848      \repeat
1849      \ifl@dpagfull
1850      \@writelinesonpageR{\the\numpagelinesR}%
1851      \else
1852      \@writelinesonpageR{1000}%
1853      \fi
1854      \numpagelinesR=\z@

```

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

```

1854      \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.

```

1855      \checkraw@text
1856      \ifaraw@text
1857      \getlinesfrompagelistL
1858      \getlinesfrompagelistR
1859      \l@dcalc@minoftwo{\@cs@linesonpageL}{\@cs@linesonpageR}%
1860      {\l@dminpagelines}%
1861      \fi
1862      \repeat}

```

We have now output the text from all the chunks.

```
1863 \fi
```

Make sure that there are no inserts hanging around.

```
1864 \flush@notes
```

```
1865 \flush@notesR
```

```
1866 \endgroup
```

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

```
1867 \global\l@dpscL=\z@
```

```
1868 \global\l@dpscR=\z@
```

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

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

```
1871 \global\instanzaLfalse
```

```
1872 \global\instanzaRfalse
```

```
1873 \ignorespaces}
```

```
1874
```

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

```
\ledstrutR 1875 \newcommand*{\ledstrutL}{\strut}
```

```
1876 \newcommand*{\ledstrutR}{\strut}
```

```
1877
```

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

```
1878 \providecommand{\cleartoevenpage}[1][\@empty]{%
```

```
1879 \clearpage
```

```
1880 \ifodd\c@page\hbox{#1}\clearpage\fi}
```

```
1881 \newcommand*{\cleartol@devenpage}{%
```

```
1882 \ifdim\pagetotal<\topskip% on an empty page
```

```
1883 \else
```

```
1884 \clearpage
```

```
1885 \fi
```

```
1886 \ifodd\c@page\hbox{#1}\clearpage\fi}
```

```
1887 \newcommand*{\clearl@dleftpage}{%
```

```
1888 \clearpage
```

```
1889 \ifodd\c@page\else
```

```
1890 \led@err@LeftOnRightPage
```

```
1891 \hbox{#1}%
```

```
1892 \cleardoublepage
```

```
1893 \fi}
```

```
1894 \newcommand*{\clearl@drighthouse}{%
```

```
1895 \clearpage
```

```
1896 \ifodd\c@page
```

```
1897 \led@err@RightOnLeftPage
```

```
1898 \hbox{#1}%
```

```

1899 \cleartoevenpage
1900 \fi}
1901

```

`\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 1902 \newcommand*{\getlinesfromparlistL}{%
1903 \ifx\linesinpar@listL\empty
1904 \gdef\cs@linesinparL{0}%
1905 \else
1906 \glp\linesinpar@listL\to\cs@linesinparL
1907 \fi}
1908 \newcommand*{\getlinesfromparlistR}{%
1909 \ifx\linesinpar@listR\empty
1910 \gdef\cs@linesinparR{0}%
1911 \else
1912 \glp\linesinpar@listR\to\cs@linesinparR
1913 \fi}
1914

```

`\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 1915 \newcommand*{\getlinesfrompagelistL}{%
1916 \ifx\linesonpage@listL\empty
1917 \gdef\cs@linesonpageL{1000}%
1918 \else
1919 \glp\linesonpage@listL\to\cs@linesonpageL
1920 \fi}
1921 \newcommand*{\getlinesfrompagelistR}{%
1922 \ifx\linesonpage@listR\empty
1923 \gdef\cs@linesonpageR{1000}%
1924 \else
1925 \glp\linesonpage@listR\to\cs@linesonpageR
1926 \fi}
1927

```

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

```

1928 \newcommand*{\@writelinesonpageL}[1]{%
1929 \edef\next{\write\linenumout{\string\@lopL{#1}}}%
1930 \next}
1931 \newcommand*{\@writelinesonpageR}[1]{%
1932 \edef\next{\write\linenumoutR{\string\@lopR{#1}}}%
1933 \next}
1934

```

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

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

```

1935 \newcommand*{\l@dcalcmisoftwo}[3]{%
1936   \ifnum #2>#1\relax
1937     #3=#2\relax
1938   \else
1939     #3=#1\relax
1940   \fi}
1941 \newcommand*{\l@dcalcmisoftwo}[3]{%
1942   \ifnum #2<#1\relax
1943     #3=#2\relax
1944   \else
1945     #3=#1\relax
1946   \fi}
1947
```

`\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 1948 \newif\ifl@dsamepage
\checkpageR 1949   \l@dsamepagetrue
1950 \newif\ifl@dpagetrue
1951 \newcommand*{\checkpageL}{%
1952   \l@dpagetrue
1953   \l@dsamepagetrue
1954   \check@goal
1955   \ifdim\pagetotal<\ledthegoal
1956     \ifnum\mypagelinesL<\l@dmypagelines
1957       \else
1958         \l@dsamepagefalse
1959         \l@dpagetruefalse
1960       \fi
1961     \else
1962       \l@dsamepagefalse
1963       \l@dpagetrue
1964     \fi}
1965 \newcommand*{\checkpageR}{%
1966   \l@dpagetrue
1967   \l@dsamepagetrue
1968   \check@goal
1969   \ifdim\pagetotal<\ledthegoal
1970     \ifnum\mypagelinesR<\l@dmypagelines
1971       \else
1972         \l@dsamepagefalse
1973         \l@dpagetruefalse
1974       \fi

```

```

1975 \else
1976     \l@dsamepagefalse
1977     \l@dpagefulltrue
1978 \fi}
1979

```

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

```

1980 \newdimen\ledthegoal
1981 \ifshiftedpstarts
1982     \newcommand*{\goalfraction}{0.95}
1983 \else
1984     \newcommand*{\goalfraction}{0.9}
1985 \fi
1986
1987 \newcommand*{\check@goal}{%
1988     \ledthegoal=\goalfraction\pagegoal}
1989

```

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

```

\ifwrittenlinesL 1990 \newif\ifwrittenlinesL
                  1991 \newif\ifwrittenlinesR
                  1992

```

`\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.

```

1993 \newcommand*{\get@nextboxL}{%
1994     \ifvbox\namebox{\l@dLcolrawbox\the\l@dpscL}% box is not empty

    The current box is not empty; do nothing.

1995     \else%                                box is empty

    The box is empty; check if enough lines (real and blank) have been output.
1996     \ifnum\usenamecount{\l@dmaxlinesinpar\the\l@dpscL}>\@donetotallinesL
1997     \else

    Sufficient lines have been output.

1998     \ifwrittenlinesL
1999     \else

```

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

```

2000     \@writelinesinparL
2001     \writtenlinesLtrue
2002 \fi
2003 \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.

```

2004      \writtenlinesLfalse
2005      \ifbypstart@
2006      \ifnum\value{pstartL}<\value{pstartLold}
2007      \else
2008      \global\line@num=0
2009      \resetprevline@
2010      \fi
2011      \fi
2012      \addtocounter{pstartL}{1}
2013      \global\pstartnumtrue
2014      \l@dcalc@maxoftwo{\the\usernamecount{1@dmaxlinesinpar\the\l@dpscL}}%
2015      {\the\@donetotallinesL}%
2016      {\usernamecount{1@dmaxlinesinpar\the\l@dpscL}}%
2017      \global\@donetotallinesL \z@
2018      \global\advance\l@dpscL \@ne
2019      \fi
2020      \fi
2021      \fi}

2022 \newcommand*{\get@nextboxR}{%
2023   \ifvbox\namebox{1@dRcolrawbox\the\l@dpscR}% box is not empty
2024   \else% box is empty
2025     \ifnum\usernamecount{1@dmaxlinesinpar\the\l@dpscR}>\@donetotallinesR
2026     \else
2027       \ifwrittenlinesR
2028       \else
2029       \@writelinesinparR
2030       \writtenlinesRtrue
2031       \fi
2032       \ifnum\l@dnumpstartsR>\l@dpscR
2033       \writtenlinesRfalse
2034       \ifbypstart@R
2035       \ifnum\value{pstartR}<\value{pstartRold}
2036       \else
2037       \global\line@numR=0
2038       \resetprevline@
2039       \fi
2040       \fi
2041       \addtocounter{pstartR}{1}
2042       \global\pstartnumRtrue
2043       \l@dcalc@maxoftwo{\the\usernamecount{1@dmaxlinesinpar\the\l@dpscR}}%
2044       {\the\@donetotallinesR}%
2045       {\usernamecount{1@dmaxlinesinpar\the\l@dpscR}}%
2046       \global\@donetotallinesR \z@
2047       \global\advance\l@dpscR \@ne
2048       \fi
2049       \fi
2050       \fi}
2051

```

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>	1505, 1506, 1510, 1527, 1541
<code>\@M</code>	1516
<code>\@adv</code>	<u>352</u> , 621, 622
<code>\@afterindentfalse</code>	744
<code>\@arabic</code>	209, 210, 793, 796
<code>\@astanza@line</code>	1526, 1532, <u>1535</u>
<code>\@auxout</code>	1316, 1328, 1615
<code>\@chapter</code>	745
<code>\@cs@linesinparL</code>	1791, <u>1902</u>
<code>\@cs@linesinparR</code>	1791, <u>1902</u>
<code>\@cs@linesonpageL</code>	1799, 1859, <u>1915</u>
<code>\@cs@linesonpageR</code>	1799, 1859, <u>1915</u>
<code>\@currentlabel</code>	828, 861
<code>\@donereallinesL</code> <u>905</u> , 934, 1753, 1755, 1805
<code>\@donereallinesR</code> <u>905</u> , 969, 1758, 1760, 1807
<code>\@donetotallinesL</code>	<u>905</u> , 935, 938, 1806, 1996, 2015, 2017
<code>\@donetotallinesR</code>	<u>905</u> , 970, 973, 1808, 2025, 2044, 2046
<code>\@insertR</code>	1270–1272, 1285–1287
<code>\@l</code>	<u>275</u> , 590
<code>\@l@dttempcnta</code>	425, 427, 429, 430, 434, 436, 438, 439, 1023, 1062, 1063, 1065, 1067, 1070, 1071, 1088–1092, 1094, 1101, 1106, 1110, 1118, 1123, 1127, 1160, 1163, 1165, 1169
<code>\@l@dttempcntb</code>	168, 170, 172, 1053, 1054, 1101, 1106, 1110, 1118, 1123, 1127, 1152, 1156, 1169, 1177–1179, 1181, 1201–1203, 1205, 1222–1224, 1226, 1357, 1359, 1361, 1432–1434, 1436, 1569–1573, 1577–1580, 1584–1587
<code>\@l@reg</code>	324
<code>\@l@regR</code>	275
<code>\@lab</code>	544, 1307, 1319, <u>1343</u>
<code>\@lock</code>	921, 1004
<code>\@lockR</code>	60, 297, 299, 301, 314, 459, 475, 476, 478, 479, 507, 508, 510, 956, 987, 1029, 1031, 1032, 1034, 1115, 1132, 1134, 1136
<code>\@lopL</code>	<u>568</u> , 1929
<code>\@lopR</code>	<u>568</u> , 1932
<code>\@nobreakfalse</code>	802, 835

- \@nobreaktrue 800, 804, 833, 837
 - \@oldnobreak 800, 802, 833, 835, 875, 891
 - \@pend 559, 1753
 - \@pendR 559, 1758
 - \@pstartsfalse 1726
 - \@pstartstrue 1726
 - \@ref 531, 594, 598
 - \@ref@reg 557
 - \@schapter 745
 - \@set 384, 628, 629, 1700, 1704
 - \@tag 658, 681, 1446
 - \@temp 1606
 - \@templ@d 1423, 1424
 - \@writelinesinparL . . 1696, 1751, 2000
 - \@writelinesinparR . . 1697, 1751, 2029
 - \@writelinesonpageL . . 1829, 1831, 1928
 - \@writelinesonpageR . . 1849, 1851, 1928
 - \@xloop 1283
- A**
- \absline@num 418, 432, 451, 995
 - \absline@numR 58, 226, 277, 280, 283,
415, 423, 444, 463, 497, 525,
536, 978, 1015, 1016, 1053, 1269
 - \actionlines@list
. 267, 270, 418, 432, 451
 - \actionlines@listR
. 230, 245, 259, 262, 415,
423, 444, 463, 497, 525, 1075, 1078
 - \actions@list 271, 419, 439, 453, 455
 - \actions@listR
. 230, 246, 263, 416, 430, 446,
448, 465, 474, 499, 506, 526, 1079
 - \add@inserts 927
 - \add@inserts@nextR 1258
 - \add@insertsR 962, 1258
 - \add@penaltiesL 933, 1279
 - \add@penaltiesR 968, 1279
 - \addtocontents 1616–1618
 - \addtocounter
. 877, 893, 1707, 1708, 2012, 2041
 - \advancelabel@refs 1314, 1326
 - \advanceline 620, 651
 - \affixline@num 925
 - \affixline@numR 960, 1085
 - \affixpstart@numL 924, 1191
 - \affixpstart@numR 959, 1191
 - \affixside@note 928
 - \affixside@noteR 963, 1410
 - \appto 1416, 1417
 - \apptocmd 1646
 - \araw@textfalse 1738
 - \araw@texttrue 1738
 - astanza (environment) 9, 1508
 - \AtBeginDocument 1339, 1591, 1626
- B**
- \ballast@count 1013, 1018
 - \bbl@main@language 1642, 1643
 - \bbl@set@language 1633, 1634
 - \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 1018
 - \c@chapter 100
 - \c@chapterR 100
 - \c@firstlinenumR 177, 1158
 - \c@firstsublinenumR 181, 1153
 - \c@linenumincrementR 177, 1158
 - \c@page 590, 1880, 1886, 1889, 1896
 - \c@pstartL 793
 - \c@pstartR 796
 - \c@section 101
 - \c@sectionR 101
 - \c@sublinenumincrementR 181, 1153
 - \c@subsection 102
 - \c@subsectionR 102
 - \c@subsubsection 103
 - \c@subsubsectionR 103
 - \ch@ck@l@ckR 1085
 - \ch@cksub@l@ckR 1085
 - \ch@cksub@lockR 1154
 - \chapter 731, 732, 740
 - \chapterinpages 724, 732, 742
 - \chardef 1505
 - \check@goal 1954, 1968, 1980
 - \check@pstarts
1671, 1698, 1726, 1785, 1793, 1801
 - \checkpageL 1812, 1826, 1948
 - \checkpageR 1835, 1846, 1948
 - \checkraw@text
. 1677, 1694, 1738, 1809, 1855
 - \cleardoublepage 1892
 - \clearl@dleftpage 1834, 1878

- `\clearl@drighthpage` 1854, [1878](#)
 - `\cleartoevenpage` [1878](#)
 - `\cleartol@evenpage` 1774, [1878](#)
 - `\closeout` 581, 585
 - `\columnrulewidth` 5, [1721](#)
 - `\Columns` 5, [1657](#)
 - `\columnseparator` 5, 1691, [1721](#)
 - `\content` 1369, 1383, 1397
 - `\correcthangingL` 930, [1486](#)
 - `\correcthangingR` 965, [1486](#)
 - `\countLline` [900](#), 911
 - `\countRline` [900](#), 946
 - `\critext` [656](#)
 - `\csexpandonce` 1372, 1376, 1386, 1390,
1400, 1404, 1448, 1452, 1462, 1467
 - `\csuse` 1461, 1466
- D**
- `\DeclareOption` 8, 9
 - `\def@tempb` 155
 - `\dimen` 607, 608, 612–614, 618
 - `\divide` 1090
 - `\do@actions` 996
 - `\do@actions@fixedcodeR` [1022](#)
 - `\do@actions@nextR` [1022](#)
 - `\do@actionsR` 979, [1022](#)
 - `\do@ballast` 997
 - `\do@ballastR` 980, [1013](#)
 - `\do@lineL` [910](#), 1681, 1685, 1817
 - `\do@lineLhook` 915, [942](#)
 - `\do@lineR` [945](#), 1682, 1687, 1838
 - `\do@lineRhook` [942](#), 950
 - `\do@lockoff` [494](#)
 - `\do@lockoffL` 518
 - `\do@lockoffR` [494](#)
 - `\do@lockon` [459](#)
 - `\do@lockonL` 491
 - `\do@lockonR` [459](#)
 - `\dolistloop` 1421
 - `\dummy@ref` 540
- E**
- `\edfont@info` 707, 710, 716, 719
 - `\edlabel` [1305](#)
 - `\edtext` [679](#)
 - `\eledmac@error` 21, 24, 27, 29
 - `\eledmac@warning` 1422
 - `\empty` .. 81, 84, 259, 267, 669, 692,
705, 714, 816, 849, 1075, 1157,
1165, 1260–1262, 1273, 1284,
1308, 1320, 1903, 1909, 1916, 1922
 - `\end@lemmas` 669, 670, 692, 693
 - `\endashchar` 1295
 - `\endgraf` 871, 887, 1665, 1777
 - `\endline@num` 547, 553
 - `\endlock` [640](#), 1514, 1523, 1528
 - `\endnumbering` 7, 39, [74](#), 121, 770
 - `\endnumberingR` 52, [74](#), 106, 116, 129, 770
 - `\endpage@num` 546, 553
 - `\endstanzaextra` 1530
 - `\endsub` [607](#)
 - `\endsubline@num` 548, 554
 - environments:
 - `astanza` 9, [1508](#)
 - `Leftside` 6, [749](#)
 - `pages` 5, [724](#)
 - `pairs` 5, [724](#)
 - `Rightside` 6, [767](#)
 - `\extensionchars` . 47, 66, 112, 126, 134
- F**
- `\f@x@l@cksR` [1085](#)
 - `\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` 1710, 1864
 - `\flush@notesR` [1282](#), 1711, 1865
 - `\fullstop` . 222, 1292, 1294, 1296, 1298
- G**
- `\get@linelistfile` 255
 - `\get@nextboxL` 1825, [1993](#)
 - `\get@nextboxR` 1845, [1993](#)
 - `\getline@numL` 920, 994
 - `\getline@numR` 955, [977](#)
 - `\getlinesfrompagelistL`
..... 1797, 1857, [1915](#)
 - `\getlinesfrompagelistR`
..... 1798, 1858, [1915](#)
 - `\getlinesfromparlistL` ... 1789, [1902](#)
 - `\getlinesfromparlistR` ... 1790, [1902](#)
 - `\gl@p` 262, 263,
270, 271, 670, 693, 709, 718,

1078, 1079, 1266, 1270, 1285,
1311, 1323, 1906, 1912, 1919, 1925
`\goalfraction` 6, 1980

H

`\hangingsymbol` 10, 1477, 1483
`\hb@xt@` ... 923, 930, 937, 958, 965,
972, 1689, 1820, 1822, 1841, 1843
`\hsize` 826,
859, 1689, 1820, 1822, 1841, 1843

I

`\if@filesw` 1614
`\if@firstcolumn` 1171, 1195, 1216, 1426
`\if@nobreak` 799, 832
`\if@pstarts` 1672, 1726, 1786, 1802
`\if@RTL` 665, 676, 688, 699
`\ifaraw@text` ... 1679, 1738, 1811, 1856
`\ifautopar` 825, 858
`\ifbypage@` 343
`\ifbypage@R` 137, 333, 1057
`\ifbypstart@` 561, 1699, 2005
`\ifbypstart@R` ... 137, 565, 1703, 2034
`\ifdim` 608, 612, 614,
618, 1820, 1841, 1882, 1955, 1969
`\iffirst@linenum@out@R` 576, 580
`\ifinserthangingsymbol` .. 1475, 1489
`\ifinserthangingsymbolR`
..... 1473, 1481, 1499
`\ifinstanzaL` 747, 747, 1476, 1488
`\ifinstanzaR` 747, 748, 1482, 1498
`\ifl@d@dash` 1295
`\ifl@d@elin` 1297, 1298
`\ifl@d@esl` 1298
`\ifl@d@pnum` 1292, 1296
`\ifl@d@ssub` 1294
`\ifl@dpagefull` 1828, 1848, 1948
`\ifl@dpadding` 11, 1487, 1497
`\ifl@dpairing` 11, 78
`\ifl@dsamelang` 1602, 1680
`\ifl@dsamepage` 1815, 1837, 1948
`\ifl@dskipnumber` 1148
`\ifl@dusedbabel` 1600
`\iflabelpstart` 828, 861
`\ifledplinum` 1293
`\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, 1306, 1344, 1358, 1371,
1385, 1399, 1447, 1460, 1637, 1647
`\ifnoteschanged@` 88
`\ifnumberedpar@` ... 809, 842, 867,
883, 1370, 1384, 1398, 1445, 1459
`\ifnumbering` 37, 142, 805, 864
`\ifnumberingR` ... 50, 75, 108, 838, 880
`\ifnumberline` 981, 998, 1147
`\ifnumberpstart` ... 825, 858, 876, 892
`\ifnumequal` 1414
`\ifnumgreater` 1422
`\ifodd` 1181, 1205,
1226, 1436, 1880, 1886, 1889, 1896
`\ifpst@rtedL` 32, 813
`\ifpst@rtedR` 32, 846
`\ifpstartnum` 1236, 1241
`\ifpstartnumR` 1191
`\ifshiftedpstarts` 5, 1819, 1840, 1981
`\ifsidepstartnum` 825, 858, 1193, 1214
`\ifsublines@` 220,
309, 353, 386, 393, 424, 433,
445, 452, 464, 498, 552, 554,
982, 999, 1064, 1151, 1346, 1350
`\ifvbox` 912, 947, 1742, 1745, 1994, 2023
`\ifvmode` 1313, 1325
`\ifwrittenlinesL` 1990, 1998
`\ifwrittenlinesR` 1991, 2027
`\initnumbering@reg` 45
`\initnumbering@sectcmd` 70
`\initnumbering@sectcountR` ... 71, 95
`\insert@count` 530, 598, 661,
684, 1378, 1392, 1406, 1454, 1469
`\insert@countR` 531, 594, 660,
683, 1374, 1388, 1402, 1450, 1464
`\inserthangingsymbolfalse` 921
`\inserthangingsymbolL` 930, 1473
`\inserthangingsymbolR` 965, 1473
`\inserthangingsymbolRfalse` 956
`\inserthangingsymbolRtrue` 956
`\inserthangingsymboltrue` 921
`\insertlines@listR`
.... 81, 230, 244, 536, 1262, 1266
`\inserts@list`
815, 1377, 1391, 1405, 1453, 1468
`\inserts@listR`
.. 848, 1257, 1260, 1270, 1284,
1285, 1373, 1387, 1401, 1449, 1463
`\instanzaLfalse` 1718, 1871
`\instanzaLtrue` 758

- \instanzaRfalse 1719, 1872
- \instanzaRtrue 780
- \interlinepenalty 1516
- \itemcount@ 1412, 1414, 1419, 1422
- L**
- \l@d@nums 707, 710, 716, 719
- \l@d@set 401, 636, 637
- \l@dbbl@set@language 1611, 1634
- \l@dbfnote 1444
- \l@dc@maxchunks 821, 823,
854, 856, 1559, 1569, 1577, 1584
- \l@dcalc@maxoftwo
..... 1791, 1935, 2014, 2043
- \l@dcalc@minoftwo .. 1799, 1859, 1935
- \l@dcalcnun 1085
- \l@dchecklang 1604, 1678
- \l@dchset@num 276, 279, 401
- \l@dcsnote 1367
- \l@dcsnotetext 1421, 1424
- \l@emptyd@ta 916, 951
- \l@dend@stuff ... 48, 67, 113, 127, 135
- \l@dgetline@margin 167
- \l@dgetsidenote@margin 1356
- \l@dld@ta 926, 961,
1172, 1184, 1196, 1208, 1217, 1229
- \l@dleftbox
.. 897, 922, 937, 1690, 1820, 1822
- \l@dlinenumR 212
- \l@dlsn@te 929, 964
- \l@dlsnote 1367
- \l@dmake@labels 1329
- \l@dmake@labelsR 1317, 1333
- \l@dminpagelines
.... 1762, 1800, 1860, 1956, 1970
- \l@dnumpstartsL . 41, 820, 821, 823,
825, 1563, 1595, 1660, 1661,
1715, 1729, 1771, 1772, 1869, 2003
- \l@dnumpstartsR . 54, 853, 854, 856,
858, 1563, 1596, 1660, 1661,
1716, 1732, 1771, 1772, 1870, 2032
- \l@doldbbl@set@language 1633
- \l@doldselectlanguage 1632, 1636, 1641
- \l@dpagefullfalse 1948
- \l@dpagefulltrue 1948
- \l@dpagingfalse 13, 726, 739
- \l@dpagingtrue 734
- \l@dpairingfalse 11, 728, 738
- \l@dpairingtrue 725, 733
- \l@dpscL 912, 917, 1565, 1597, 1669,
1675, 1713, 1729, 1742, 1781,
1787, 1792, 1795, 1803, 1867,
1994, 1996, 2003, 2014, 2016, 2018
- \l@dpscR 947, 952, 1566, 1598,
1670, 1676, 1714, 1732, 1745,
1782, 1788, 1796, 1804, 1868,
2023, 2025, 2032, 2043, 2045, 2047
- \l@drd@ta 930, 965,
1174, 1182, 1198, 1206, 1219, 1227
- \l@drightbox
.. 897, 957, 972, 1692, 1841, 1843
- \l@drrsn@te 931, 966
- \l@drrsnote 1367
- \l@dsamelangfalse 1602, 1605
- \l@dsamelangtrue 1602, 1608
- \l@dsamepagefalse 1948
- \l@dsamepagetrue 1948
- \l@dsetupmaxlinecounts .. 1576, 1593
- \l@dsetupprawboxes 1568, 1592
- \l@dskipnumberfalse 1149
- \l@dskipnumbertrue 1045
- \l@dunhbox@line 930, 965
- \l@dusedbafelfalse 1600, 1629
- \l@dusedbabeltrue 1600, 1631
- \l@duselanguage
.... 1621, 1684, 1686, 1813, 1836
- \l@dzeromaxlinecounts ... 1576, 1594
- \l@dzeropenalties 870, 886, 1664, 1776
- \l@pscl 1565
- \l@pscr 1565
- \label@refs
1309, 1311, 1317, 1321, 1323, 1329
- \labelref@list 1320, 1323, 1351
- \labelref@listR 1303, 1308, 1311, 1347
- \languagenam .. 1612, 1613, 1615–1618
- \last@page@num 341, 347
- \last@page@numR 327
- \lastbox 919, 954
- \lastskip 607, 613
- \Lcolwidth 5, 6, 15, 735, 826, 923, 937
- \led@err@BadLeftRightPstarts ...
..... 23, 1661, 1772
- \led@err@LeftOnRightPage ... 26, 1890
- \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, 1897
 - \led@err@TooManyPstarts 20, 822, 855
 - \led@mess@NotesChanged 89
 - \led@mess@SectionContinued
 - 111, 125, 133
 - \led@warn@BadAction 1047
 - \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 1335
 - \ledllfill 930, 965
 - \ledRcolfalse 14, 750, 782
 - \ledRcoltrue 768
 - \ledrlfill 930, 965
 - \ledsavedprintlines 8, 1290
 - \ledstrutL 1820, 1822, 1875
 - \ledstrutR 1841, 1843, 1875
 - \ledthegoal 1955, 1969, 1980
 - \leftlinenumR 212, 1172, 1184
 - \leftpstartnumL 1191
 - \leftpstartnumR 1191
 - 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, 1201
 - \line@marginR 165, 1177, 1222
 - \line@num . 344, 375, 376, 378, 396,
 - 407, 408, 436, 561, 1005, 1349, 2008
 - \line@numR 59, 219,
 - 226, 281, 315, 334, 369, 370,
 - 372, 389, 403, 404, 427, 547,
 - 551, 565, 988, 1058, 1067, 1156,
 - 1158, 1160, 1161, 1345, 1422, 2037
 - \lineation 777
 - \lineationR 141, 777
 - \linenum@out 597,
 - 605, 610, 616, 622, 629, 637,
 - 642, 646, 1319, 1700, 1753, 1929
 - \linenum@outR 575, 581,
 - 583, 585, 586, 590, 593, 603,
 - 609, 615, 621, 628, 636, 641,
 - 645, 650, 1307, 1704, 1758, 1932
 - \linenumberlist 1157, 1161
 - \linenumincrement 6, 186
 - \linenummargin 165
 - \linenumr@p 1293, 1297, 1345, 1349
 - \linenumrepR 209, 219
 - \linenumsep
 - . 214, 216, 1238, 1241, 1250, 1253
 - \linesinpar@listL
 - 235, 251, 562, 1903, 1906
 - \linesinpar@listR
 - 235, 247, 566, 1909, 1912
 - \linesonpage@listL 252, 570, 1916, 1919
 - \linesonpage@listR 248, 573, 1922, 1925
 - \list@clear
 - . 243–248, 251, 252, 254, 815, 848
 - \list@clearing@reg 250
 - \list@create
 - ... 230–233, 235–237, 1257, 1303
 - \lock@disp 1117, 1121, 1126
 - \lock@off 485, 486, 494, 645, 646
 - \lock@on 641, 642
- M
- \maxchunks 4, 1559
 - \maxlinesinpar@list 235, 254
 - \memorydump 8, 755, 773
 - \memorydumpL 120, 755
 - \memorydumpR 120, 773
 - \message 46, 65
 - \multiply 1091
- N
- \n@num 522, 650
 - \n@num@reg 528
 - \namebox 912, 917, 947,
 - 952, 1543, 1742, 1745, 1994, 2023
 - \NeedsTeXFormat 2
 - \new@line 930
 - \new@lineR 589, 965
 - \newbox 788, 897, 898, 1544
 - \newcounter 95–98, 177,
 - 179, 181, 183, 791, 792, 794, 795
 - \newif . 5, 12, 33, 137, 138, 576, 747,
 - 748, 1245, 1473, 1600, 1602,
 - 1726, 1738, 1948, 1950, 1990, 1991
 - \newnamebox 1543, 1571, 1572
 - \newnamecount 1554, 1579
 - \newwrite 575

- \next@action 271
 - \next@actionline 268, 270
 - \next@actionlineR
 - . 260, 262, 1016, 1054, 1076, 1078
 - \next@actionR 263, 1017,
 - 1055, 1056, 1061, 1062, 1070, 1079
 - \next@insert 816
 - \next@insertR
 - 849, 1261, 1264, 1266, 1269, 1273
 - \next@page@num 348, 419
 - \next@page@numR 63, 284, 286, 338, 416
 - \no@expands 658, 681
 - \normal@pars 77, 819, 852
 - \normalbfnoteX 1458
 - \noteschanged@true
 - 82, 85, 706, 715, 1263
 - \num@lines 871, 1665, 1777
 - \num@linesR 787, 887, 1666, 1778
 - \numberedpar@true 827, 860
 - \numberingRfalse 76
 - \numberingRtrue 56, 106, 130
 - \numberingtrue 43, 122
 - \numberpstartfalse 9
 - \numberpstarttrue 9
 - \numdef 1412, 1419
 - \numlabfont 219
 - \numpagelinesL
 - 1762, 1818, 1829, 1833, 1956
 - \numpagelinesR
 - 1762, 1839, 1849, 1853, 1970
- O**
- \oldchapter 731, 740
 - \oldstanza 757, 758, 760, 779, 780, 783
 - \one@line 917, 919, 930
 - \one@lineR 787, 952, 954, 965
 - \openout 583, 586
- P**
- \p@pstartL 829
 - \p@pstartR 862
 - \page@action 285, 413, 541
 - \page@num 266, 346, 1203, 1434
 - \page@numR 239, 258, 336,
 - 546, 551, 1056, 1179, 1224, 1422
 - \pagegoal 1988
 - \Pages 5, 1766
 - pages (environment) 5, 724
 - \pagetotal 1882, 1955, 1969
 - pairs (environment) 5, 724
 - \par@line 872, 1667, 1779
 - \par@lineR 787, 888, 1668, 1780
 - \pausenumbering 771
 - \pausenumberingR 105, 771
 - \pend 6, 754, 776, 811, 1529
 - \pendL 754, 864
 - \pendR 776, 844, 880
 - \prevgraf
 - . 871, 887, 1665, 1666, 1777, 1778
 - \printlines 1301
 - \printlinesR 8, 1290
 - \ProcessOptions 10
 - \protected@csxdef 1461, 1466
 - \protected@edef 828, 861
 - \protected@write ... 1316, 1328, 1615
 - \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, 1531
 - \pstartL 752, 790
 - \pstartnumfalse 1238, 1243
 - \pstartnumRfalse 1250, 1255
 - \pstartnumRtrue 1246, 1674, 2042
 - \pstartnumtrue 1673, 2013
 - \pstartR 775, 790
- R**
- \Rcolwidth 5, 6, 15, 736, 859, 958, 972
 - \read@linelist 241, 579
 - \rem@inder 1161, 1163–1165
 - \resetprevline@ 1701, 1705, 2009, 2038
 - \resumenumbering 772
 - \resumenumberingR 105, 772
 - \rightlinenumR 212, 1174, 1182
 - \rightpstartnumL 1191
 - \rightpstartnumR 1191
 - Rightside (environment) 6, 767
 - \Rightsidehook 762, 778
 - \Rightsidehookend 762, 784
 - \rlap 1174, 1182, 1198, 1206, 1219, 1227
 - \Rlineflag 8, 207, 219, 1293, 1297, 1337
 - \rule 1722
- S**
- \sc@n@list 1162, 1164
 - \secdef 745
 - \section@num 44, 46, 47, 124–126

- \section@numR 30, 57, 65, 66, 110–112, 132–134
 - \select@language ... 1613, 1615–1618
 - \selectlanguage 1621
 - \set@line 659, 682, 703
 - \set@line@action 278, 382, 391, 398, 421, 543
 - \setl@dlp@rbox 1427, 1439
 - \setl@drp@rbox 1429, 1437
 - \setline 624
 - \setlinenum 632
 - \setnamebox 825, 858, 1543
 - \setprintlines 1291
 - \shiftedpstartsfalse 7
 - \shiftedpstartstrue 6, 8, 9
 - \shiftedversesfalse 7
 - \shiftedversestrue 6
 - \showlemma 668, 691
 - \sidenote@margin 1361, 1365
 - \sidenote@marginR 1354, 1432
 - \sidenotecontent@ 1411, 1416, 1417, 1427, 1429, 1439
 - \sidenotecontent@t 1437
 - \sidenotemargin 1354
 - \sidenotesep 1417
 - \skip@lockoff 486, 494
 - \skipnumbering 9, 649
 - \skipnumbering@reg 653
 - \smash 1722
 - \splittopskip 914, 949
 - \stanza ... 757, 758, 760, 779, 780, 783
 - \stanza@count 1511, 1525, 1536
 - \stanza@hang 1513, 1538
 - \stanzaindentbase .. 1491, 1501, 1536
 - \startlock 640
 - \startstanzahook 1509
 - \startsub 607
 - \sub@action 294, 442, 542
 - \sub@change 64, 288, 289, 295
 - \sub@lock 1000
 - \sub@lockR 61, 303, 305, 307, 310, 460, 466, 467, 469, 470, 500, 501, 503, 983, 1037, 1039, 1040, 1042, 1098, 1138, 1140, 1142
 - \sub@off 615, 616
 - \sub@on 609, 610
 - \subline@num 221, 344, 361, 362, 364, 394, 434, 1001, 1006, 1350
 - \subline@numR 222, 226, 311, 315, 334, 355, 356, 358, 387, 425, 548, 552, 984, 989, 1058, 1065, 1152, 1153, 1346
 - \sublinenumincrement 6, 186
 - \sublinenumr@p . 1294, 1298, 1346, 1350
 - \sublinenumrepR 209, 222
 - \sublines@false 62, 292, 1027
 - \sublines@true 290, 1025
 - \sublock@disp 1100, 1104, 1109
 - \sympplinenum 1293
 - \sza@penalty 1520, 1524
- T**
- \textwidth 16, 18, 735, 736
 - \theledlanguageL 1606, 1621, 1684, 1813
 - \theledlanguageR 1606, 1621, 1686, 1836
 - \thepage 590, 1317, 1329
 - \thepstart 753, 774
 - \thepstartL 9, 753, 793, 825, 829, 1237, 1242
 - \thepstartR 9, 774, 796, 858, 862, 1249, 1254
 - \thr@@ .. 469, 478, 501, 508, 1032, 1040
 - \topskip 1882
- U**
- \unhbox 1548, 1690, 1692, 1820, 1822, 1841, 1843
 - \unhnamebox 1543
 - \unvbox 919, 954, 1550
 - \unvnamebox 1543
 - \usernamecount 1512, 1519, 1554, 1586, 1792, 1996, 2014, 2016, 2025, 2043, 2045
- V**
- \value 814, 847, 1658, 1659, 1767, 1768, 2006, 2035
 - \vbadness 913, 948
 - \vbfnoteX 1462, 1467
 - \vbox 825, 858
 - \vl@dbfnote 1448, 1452
 - \vl@dcsnote 1400, 1404
 - \vl@dlsnote 1372, 1376
 - \vl@drsnote 1386, 1390
 - \vsplit 917, 952
- W**
- \wd 930, 965
 - \writtenlinesLfalse 1783, 2004
 - \writtenlinesLtrue 2001

<code>\writtenlinesRfalse</code>	1784, 2033	430, 432, 439, 444, 446, 448,
<code>\writtenlinesRtrue</code>	2030	451, 453, 455, 463, 465, 474,
X		497, 499, 506, 525, 526, 536,
<code>\x@lemma</code>	670–672, 693–695	550, 562, 566, 570, 573, 1345,
<code>\xpg@main@language</code>	1651, 1652	1349, 1372, 1376, 1386, 1390,
<code>\xpg@set@language</code>	1646, 1650	1400, 1404, 1448, 1452, 1462, 1467
<code>\xright@appenditem</code>		Z
.	415, 416, 418, 419, 423,	
		<code>\zz@@@</code>
		1309, 1321

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@dcchecklang</code> and <code>\l@duselanguage</code> to <code>\Columns</code>	60
New management of hangingsymbol insertion, preventing undesirable insertions.	53	<code>\Pages</code> : Added <code>\l@duselanguage</code> to <code>\Pages</code>	64
Prevent shift of column separator when a verse is hanged	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: Reorganize for ledarab	1
<code>\Columns</code> : Line numbering by pstart.	61	<code>\affixline@numR</code> : Changed <code>\affixline@numR</code> to match new eledmac	43
<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>\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
		<code>\do@lineRhook</code> : Added <code>\do@lineLhook</code> and <code>\do@lineRhook</code>	40

Leftside: Added hooks into Left-side environment	34	v0.5	General: Corrections about \section and other titles in numbered sections	1
\flag@end: Removed extraneous spaces from \flag@end	29			
\ifledRcol: Moved \ifl@dpairing to eledmac	13	v0.6	General: Be able to use \chapter in parallel pages.	1
\ifpst@rtedR: Moved \ifpst@rtedL to eledmac	14	v0.7	General: Option ‘shiftedverses’ which make there is no blank between two parallel verses with inequal length.	1
\l@dlinenumR: Simplified \leftlinenumR and \rightlinenumR by introducing \l@dlinenumR	19			
\l@dnumpstartsR: Moved \l@dnumpstartsL to eledmac	56	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
\ledsavedprintlines: Simplified \setprintlines	49			
\ledstrutR: Added \ledstrutL and \ledstrutR	66			
\normalbfnoteX: Removed extraneous spaces from \normalbfnoteX	53	v0.9	General: Possibility to number \pstart.	9
\Pages: Added \ledstrutL to \Pages	64		Possibility to number the pstart with the commands \numberpstarttrue.	1
\Rightsidehookend: Added \Leftsidehook, \Leftsidehookend, \Rightsidehook and \Rightsidehookend	34	v0.9.1	\ifledRcol: Moved \iflledRcol and \ifnumberingR to eledmac	13
\sublinenumrepR: Added \linenumrepR and \sublinenumrepR	19	v0.9.2	General: The numbering of the pstarts restarts on each \beginnumbering.	1
v0.3a				
General: Minor \linenummargin fix	1		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
\line@marginR: Don’t just set \line@marginR in \linenummargin	17			
v0.3b				
General: Improved parallel page balancing	1	v0.9.3	General: \thepstartL and \thepstartR use now \bfseries and not \bf, which is deprecated and makes conflicts with memoir class.	1
\Pages: Added \l@dminpagelines calculation for succeeding page pairs	65			
v0.3c				
General: Compatibility with Polyglossia	1	v1.0	General: Compatibility with eledmac. Change name to eledpar.	1
v0.4			Debug in lineation by pstart	16
General: No more ledparpatch. All patches are now in the main file.	1	v1.0.1	General: Correction on \numberonlyfirstinline with lineation by pstart or by	

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