

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

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

Abstract

The **eledmac** package, which is based on the PLAIN T_EX set of **EDMAC** macros, has been used for some time for typesetting critical editions. The **eledpar** package is an extension to **eledmac** which enables texts and their critical apparatus to be typeset in parallel, either in two columns or on pairs of facing pages.

Note that before September 2012, **eledpar** was called **ledpar**. The changes from **ledmac/ledpar** to **eledmac/eledpar** is explained in **ledmac** documentation.

To report bugs, please go to **ledmac**'s GitHub page and click “New Issue”: <https://github.com/maieul/ledmac/issues/>. You must open an account with github.com to access my page (maieul/ledmac). GitHub accounts are free for open-source users.

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

Contents

1	Introduction	3
2	The eledpar package	3
2.1	General	4
3	Parallel columns	5
4	Facing pages	5
5	Left and right texts	6
6	Numbering text lines and paragraphs	7

*This file (**eledpar.dtx**) has version number v1.1, last revised 2012/09/25.

[†]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	28
12 Marking text for notes	31
13 Parallel environments	32
14 Paragraph decomposition and reassembly	34
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	42
14.5 Pstart number printing in side	44
14.6 Add insertions to the vertical list	46
14.7 Penalties	47
14.8 Printing leftover notes	48
15 Footnotes	48
15.1 Normal footnote formatting	48
16 Cross referencing	48
17 Side notes	50
18 Familiar footnotes	52
19 Verse	52
20 Naming macros	54
21 Counts and boxes for parallel texts	55
22 Fixing babel	56
23 Parallel columns	58

<i>List of Figures</i>	3
24 Parallel pages	61
25 The End	70
References	71
Index	71
Change History	79

List of Figures

1 Introduction

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

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

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

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

2 The **eledpar** package

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

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

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

2.1 General

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

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

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

```
\maxchunks{5120}
```

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

TeX has a limited number of boxes; if you get an error message along the lines of ‘no room for a new box’, then load the package *etex*, which needs *pdf_latex* or *x_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}[2012/09/25 v1.1 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 }
70

```

`\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.

```

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

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

```

\resumenumberingR 92 \newcommand*{\pausenumberingR}{%
93   \endnumberingR\global\numberingRtrue}
94 \newcommand*{\resumenumberingR}{%
95   \ifnumberingR
96     \global\pst@rtedRtrue
97     \global\advance\section@numR \@ne
98     \led@mess@SectionContinued{\the\section@numR R}%
99     \line@list@stuffR{\jobname.\extensionchars\the\section@numR R}%
100    \l@dend@stuff
101  \else
102    \led@err@numberingShouldHaveStarted
103    \endnumberingR
104    \beginnumberingR
105  \fi}
106
```

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

```

107 \newcommand*{\memorydumpL}{%
108   \endnumbering
```

```

109 \numberingtrue
110 \global\pst@rtedLtrue
111 \global\advance\section@num \@ne
112 \led@mess@SectionContinued{\the\section@num}%
113 \line@list@stuff{\jobname.\extensionchars\the\section@num}%
114 \l@dend@stuff}
115 \newcommand*\memorydumpR}{%
116 \endnumberingR
117 \numberingRtrue
118 \global\pst@rtedRtrue
119 \global\advance\section@numR \@ne
120 \led@mess@SectionContinued{\the\section@numR R}%
121 \line@list@stuffR{\jobname.\extensionchars\the\section@numR R}%
122 \l@dend@stuff}
123

```

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:

- line-of-page : `bypstart@R = false` and `bypage@R = true`.
- line-of-pstart : `bypstart@R = true` and `bypage@R = false`.

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

```

124 \newif\ifbypage@R
125 \newif\ifbypstart@R
126 \bypage@Rfalse
127 \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`.

```

128 \newcommand*\lineationR[1]{%
129 \ifnumbering
130 \led@err@LineationInNumbered
131 \else
132 \def\@tempa{#1}\def\@tempb{page}%
133 \ifx\@tempa\@tempb
134 \global\bypage@Rtrue
135 \global\bypstart@Rfalse
136 \else

```



```

137     \def\@tempb{pstart}%
138     \ifx\@tempa\@tempb
139         \global\bypage@Rfalse
140         \global\bystart@Rtrue
141     \else
142         \def\@tempb{section}
143         \ifx\@tempa\@tempb
144             \global\bypage@Rfalse
145             \global\bystart@Rfalse
146         \else
147             \led@warn@BadLineation
148         \fi
149     \fi
150 \fi
151 \fi}}

```

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

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

```

152 \newcount\line@marginR
153 \renewcommand*{\linenummargin}[1]{%
154     \l@getline@margin{#1}%
155     \ifnum\@l@tempcntb>\m@ne
156         \ifledRcol
157             \global\line@marginR=\@l@tempcntb
158         \else
159             \global\line@margin=\@l@tempcntb
160         \fi
161     \fi}}

```

By default put right text numbers at the right.

```

162 \line@marginR=\@ne
163

```

`\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.

```

164 \newcounter{firstlinenumR}
165 \setcounter{firstlinenumR}{5}
166 \newcounter{linenumincrementR}
167 \setcounter{linenumincrementR}{5}

```

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

```
168 \newcounter{firstsublinenumR}
169 \setcounter{firstsublinenumR}{5}
170 \newcounter{sublinenumincrementR}
171 \setcounter{sublinenumincrementR}{5}
172
```

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

```
\firstsublinenum 173 \providecommand*\firstlinenum{}
\sublinenumincrement 174 \providecommand*\linenumincrement{}
175 \providecommand*\firstsublinenum{}
176 \providecommand*\sublinenumincrement{}
177 \renewcommand*\firstlinenum[1]{%
178   \ifledRcol \setcounter{firstlinenumR}{#1}%
179   \else      \setcounter{firstlinenumR}{#1}%
180   \fi}
181 \renewcommand*\linenumincrement[1]{%
182   \ifledRcol \setcounter{linenumincrementR}{#1}%
183   \else      \setcounter{linenumincrementR}{#1}%
184   \fi}
185 \renewcommand*\firstsublinenum[1]{%
186   \ifledRcol \setcounter{firstsublinenumR}{#1}%
187   \else      \setcounter{firstsublinenumR}{#1}%
188   \fi}
189 \renewcommand*\sublinenumincrement[1]{%
190   \ifledRcol \setcounter{sublinenumincrementR}{#1}%
191   \else      \setcounter{sublinenumincrementR}{#1}%
192   \fi}
193
```

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

```
194 \newcommand*\Rlineflag{R}
195
```

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

```
196 \newcommand*\linenumrepR[1]{\@arabic{#1}}
197 \newcommand*\sublinenumrepR[1]{\@arabic{#1}}
198
```

`\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`.

```
199 \newcommand*\leftlinenumR{%
200   \l@dlinenumR
201   \kern\linenumsep}
```

```

202 \newcommand*{\rightlinenumR}{%
203   \kern\linenumsep
204   \l@dlinenumR}
205 \newcommand*{\l@dlinenumR}{%
206   \numlabfont\linenumrepR{\line@numR}\Rlineflag%
207   \ifsublines@
208     \ifnum\subline@num>\z@
209       \unskip\fullstop\sublinenumrepR{\subline@numR}%
210     \fi
211   \fi}
212

```

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 marginal line numbering and in notes. The count `\subline@numR` stores a sub-line 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.

```

213 \newcount\line@numR
214 \newcount\subline@numR
215 \newcount\absline@numR
216

```

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

`\actions@listR` Here are the commands to create these lists:

```

217 \list@create{\line@listR}
218 \list@create{\insertlines@listR}
219 \list@create{\actionlines@listR}
220 \list@create{\actions@listR}
221

```

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

```

222 \list@create{\linesinpar@listL}
223 \list@create{\linesinpar@listR}
224 \list@create{\maxlinesinpar@list}
225

```

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

```

226 \newcount\page@numR
227

```

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.

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

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

```
229 \ifledRcol
230 \list@clear{\line@listR}%
231 \list@clear{\insertlines@listR}%
232 \list@clear{\actionlines@listR}%
233 \list@clear{\actions@listR}%
234 \list@clear{\linesinpar@listR}%
235 \list@clear{\linesonpage@listR}%
236 \else
237 \list@clearing@reg
238 \list@clear{\linesinpar@listL}%
239 \list@clear{\linesonpage@listL}%
240 \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).

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

Now get the file and interpret it.

```
242 \get@linelistfile{#1}%
243 \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.

```
244 \ifledRcol
245 \global\page@numR=\m@ne
246 \ifx\actionlines@listR\empty
247 \gdef\next@actionlineR{1000000}%
248 \else
249 \gl@p\actionlines@listR\to\next@actionlineR
250 \gl@p\actions@listR\to\next@actionR
251 \fi
252 \else
253 \global\page@num=\m@ne
254 \ifx\actionlines@list\empty
255 \gdef\next@actionline{1000000}%
256 \else
257 \gl@p\actionlines@list\to\next@actionline
258 \gl@p\actions@list\to\next@action
259 \fi
260 \fi}
261
```

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.

```

262 \newcommand{\@l@regR}{%
263   \ifx\l@dchset@num\relax \else
264     \advance\absline@numR \@ne
265     \set@line@action
266     \let\l@dchset@num\relax
267     \advance\absline@numR \m@ne
268     \advance\line@numR \m@ne%    % do we need this?
269   \fi
270   \advance\absline@numR \@ne
271   \ifx\next@page@numR\relax \else
272     \page@action
273     \let\next@page@numR\relax
274   \fi
275   \ifx\sub@change\relax \else
276     \ifnum\sub@change>\z@
277       \sublines@true
278     \else
279       \sublines@false
280     \fi
281     \sub@action
282     \let\sub@change\relax
283   \fi
284   \ifcase\@lockR
285   \or
286     \@lockR \tw@
287   \or\or
288     \@lockR \z@
289   \fi
290   \ifcase\sub@lockR
291   \or
292     \sub@lockR \tw@
293   \or\or
294     \sub@lockR \z@
295   \fi
296   \ifsublines@

```

```

297 \ifnum\sub@lockR<\tw@
298 \advance\subline@numR \@ne
299 \fi
300 \else
301 \ifnum\@lockR<\tw@
302 \advance\line@numR \@ne \subline@numR \z@
303 \fi
304 \fi}
305
306 \renewcommand*{\@l}[2]{%
307 \fix@page{#1}%
308 \ifledRcol
309 \@l@regR
310 \else
311 \@l@reg
312 \fi}
313

```

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

```

\fix@page 314 \newcount\last@page@numR
315 \last@page@numR=-10000
316 \renewcommand*{\fix@page}[1]{%
317 \ifledRcol
318 \ifnum #1=\last@page@numR
319 \else
320 \ifbypage@R
321 \line@numR \z@ \subline@numR \z@
322 \fi
323 \page@numR=#1\relax
324 \last@page@numR=#1\relax
325 \def\next@page@numR{#1}%
326 \fi
327 \else
328 \ifnum #1=\last@page@num
329 \else
330 \ifbypage@
331 \line@num \z@ \subline@num \z@
332 \fi
333 \page@num=#1\relax
334 \last@page@num=#1\relax
335 \def\next@page@num{#1}%
336 \fi
337 \fi}
338

```

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

```

339 \renewcommand*{\@adv}[1]{%
340 \ifsublines@

```

```

341 \ifledRcol
342 \advance\subline@numR by #1\relax
343 \ifnum\subline@numR<\z@
344 \led@warn@BadAdvancelineSubline
345 \subline@numR \z@
346 \fi
347 \else
348 \advance\subline@num by #1\relax
349 \ifnum\subline@num<\z@
350 \led@warn@BadAdvancelineSubline
351 \subline@num \z@
352 \fi
353 \fi
354 \else
355 \ifledRcol
356 \advance\line@numR by #1\relax
357 \ifnum\line@numR<\z@
358 \led@warn@BadAdvancelineLine
359 \line@numR \z@
360 \fi
361 \else
362 \advance\line@num by #1\relax
363 \ifnum\line@num<\z@
364 \led@warn@BadAdvancelineLine
365 \line@num \z@
366 \fi
367 \fi
368 \fi
369 \set@line@action}
370

```

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

```

371 \renewcommand*{\@set}[1]{%
372 \ifledRcol
373 \ifsublines@
374 \subline@numR=#1\relax
375 \else
376 \line@numR=#1\relax
377 \fi
378 \set@line@action
379 \else
380 \ifsublines@
381 \subline@num=#1\relax
382 \else
383 \line@num=#1\relax
384 \fi
385 \set@line@action
386 \fi}
387

```

`\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 line number change is to be done.

```

388 \renewcommand*{\l@d@set}[1]{%
389   \ifl@dchset@num
390     \line@numR=#1\relax
391     \advance\line@numR \@ne
392     \def\l@dchset@num{#1}
393   \else
394     \line@num=#1\relax
395     \advance\line@num \@ne
396     \def\l@dchset@num{#1}
397   \fi}
398 \let\l@dchset@num\relax
399

```

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

```

400 \renewcommand*{\page@action}{%
401   \ifl@dchset@num
402     \xright@appenditem{\the\absline@numR}\to\actionlines@listR
403     \xright@appenditem{\next@page@numR}\to\actions@listR
404   \else
405     \xright@appenditem{\the\absline@num}\to\actionlines@list
406     \xright@appenditem{\next@page@num}\to\actions@list
407   \fi}

```

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

```

408 \renewcommand*{\set@line@action}{%
409   \ifl@dchset@num
410     \xright@appenditem{\the\absline@numR}\to\actionlines@listR
411     \ifsublines@
412       \l@dtempcnta=-\subline@numR
413     \else
414       \l@dtempcnta=-\line@numR
415     \fi
416     \advance\l@dtempcnta by -5000\relax
417     \xright@appenditem{\the\l@dtempcnta}\to\actions@listR
418   \else
419     \xright@appenditem{\the\absline@num}\to\actionlines@list
420     \ifsublines@
421       \l@dtempcnta=-\subline@num
422     \else
423       \l@dtempcnta=-\line@num
424     \fi
425     \advance\l@dtempcnta by -5000\relax
426     \xright@appenditem{\the\l@dtempcnta}\to\actions@list
427   \fi}

```


428

`\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.

```

429 \renewcommand*{\sub@action}{%
430   \ifledRcol
431     \xright@appenditem{\the\absline@numR}\to\actionlines@listR
432     \ifsublines@
433       \xright@appenditem{-1001}\to\actions@listR
434     \else
435       \xright@appenditem{-1002}\to\actions@listR
436     \fi
437   \else
438     \xright@appenditem{\the\absline@num}\to\actionlines@list
439     \ifsublines@
440       \xright@appenditem{-1001}\to\actions@list
441     \else
442       \xright@appenditem{-1002}\to\actions@list
443     \fi
444   \fi}
445
```

`\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.

```

446 \newcount\@lockR
447 \newcount\sub@lockR
448
449 \newcommand*{\do@lockonR}{%
450   \xright@appenditem{\the\absline@numR}\to\actionlines@listR
451   \ifsublines@
452     \xright@appenditem{-1005}\to\actions@listR
453     \ifnum\sub@lockR=\z@
454       \sub@lockR \@ne
455     \else
456       \ifnum\sub@lockR=\thr@@
457         \sub@lockR \@ne
458       \fi
459     \fi
460   \else
461     \xright@appenditem{-1003}\to\actions@listR
462     \ifnum\@lockR=\z@
463       \@lockR \@ne
464     \else
465       \ifnum\@lockR=\thr@@
466         \@lockR \@ne
467       \fi
468     \fi
469   \fi}

```

```

470
471 \renewcommand*{\do@lockon}{%
472   \ifx\next\lock@off
473     \global\let\lock@off=\skip@lockoff
474   \else
475     \ifledRcol
476       \do@lockonR
477     \else
478       \do@lockonL
479     \fi
480   \fi}

\lock@off \lock@off adds an entry to the action-code list to turn line number locking off.
\do@lockoff 481
\do@lockoffR 482
\skip@lockoff 483 \newcommand{\do@lockoffR}{%
484   \xright@appenditem{\the\absline@numR}\to\actionlines@listR
485   \ifsublines@
486     \xright@appenditem{-1006}\to\actions@listR
487     \ifnum\sub@lockR=\tw@
488       \sub@lockR \thr@@
489     \else
490       \sub@lockR \z@
491     \fi
492   \else
493     \xright@appenditem{-1004}\to\actions@listR
494     \ifnum\@lockR=\tw@
495       \@lockR \thr@@
496     \else
497       \@lockR \z@
498     \fi
499   \fi}
500
501 \renewcommand*{\do@lockoff}{%
502   \ifledRcol
503     \do@lockoffR
504   \else
505     \do@lockoffL
506   \fi}
507 \global\let\lock@off=\do@lockoff
508

\n@num This macro implements the \skipnumbering command. It uses a new action code,
namely 1007.
509 \providecommand*{\n@num}{}
510 \renewcommand*{\n@num}{%
511   \ifledRcol
512     \xright@appenditem{\the\absline@numR}\to\actionlines@listR
513     \xright@appenditem{-1007}\to\actions@listR
514   \else

```

```

515     \n@num@reg
516     \fi}
517

```

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

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

```

518     \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.

```

519 \renewcommand*{\@ref}[2]{%
520   \ifledRcol
521     \global\insert@countR=#1\relax
522     \loop\ifnum\insert@countR>\z@
523       \xright@appenditem{\the\absline@numR}\to\insertlines@listR
524       \global\advance\insert@countR \m@ne
525     \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.

```

526   \begingroup
527     \let\@ref=\dummy@ref
528     \let\page@action=\relax
529     \let\sub@action=\relax
530     \let\set@line@action=\relax
531     \let\@lab=\relax
532     #2
533     \global\endpage@num=\page@numR
534     \global\endline@num=\line@numR
535     \global\endsubline@num=\subline@numR
536   \endgroup

```

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

```

537     \xright@appenditem%
538       {\the\page@numR|\the\line@numR|}%
539       \ifsublines@ \the\subline@numR \else 0\fi|}%
540       \the\endpage@num|\the\endline@num|}%
541       \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.

```
542 #2
543 \else
```

And when not in right text

```
544 \@ref@reg{#1}{#2}%
545 \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.

```
546 \providecommand*\@pend}[1]{%
547 \renewcommand*\@pend}[1]{%
548 \ifbypstart@global\line@num=0\fi%
549 \xright@appenditem{#1}\to\linesinpar@listL}
550 \providecommand*\@pendR}[1]{%
551 \renewcommand*\@pendR}[1]{%
552 \ifbypstart@Rglobal\line@numR=0\fi
553 \xright@appenditem{#1}\to\linesinpar@listR}
554
```

`\@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.

```
555 \providecommand*\@lopL}[1]{%
556 \renewcommand*\@lopL}[1]{%
557 \xright@appenditem{#1}\to\linesonpage@listL}
558 \providecommand*\@lopR}[1]{%
559 \renewcommand*\@lopR}[1]{%
560 \xright@appenditem{#1}\to\linesonpage@listR}
561
```

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`.

```
562 \newwrite\linenum@outR
```

`\iffirst@linenum@out@R` Once any file is opened on this stream, we keep it open forever, or else switch to another file that we keep open.

```
\first@linenum@out@Rtrue
\first@linenum@out@Rfalse 563 \newif\iffirst@linenum@out@R
564 \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.

```

565 \newcommand*{\line@list@stuffR}[1]{%
566   \read@linelist{#1}%
567   \iffirst@linenum@out@R
568     \immediate\closeout\linenum@outR
569     \global\first@linenum@out@Rfalse
570     \immediate\openout\linenum@outR=#1
571   \else
572     \closeout\linenum@outR
573     \openout\linenum@outR=#1
574   \fi}
575
```

`\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.

```

576 \newcommand*{\new@lineR}{%
577   \write\linenum@outR{\string\@l[\the\c@page][\thepage]}}
```

`\flag@start` We enclose a lemma marked by `\edtext` in `\flag@start` and `\flag@end`: these
`\flag@end` send the `\@ref` command to the line-list file.

```

578 \renewcommand*{\flag@start}{%
579   \ifledRcol
580     \edef\next{\write\linenum@outR{%
581       \string\@ref[\the\insert@countR] []}%
582     \next
583   \else
584     \edef\next{\write\linenum@outR{%
585       \string\@ref[\the\insert@count] []}%
586     \next
587   \fi}
588 \renewcommand*{\flag@end}{%
589   \ifledRcol
590     \write\linenum@outR{[]}%
591   \else
592     \write\linenum@outR{[]}%
593   \fi}
```

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

```

594 \renewcommand*{\startsub}{\dimen0\lastskip
595   \ifdim\dimen0>0pt \unskip \fi
596   \ifledRcol \write\linenum@outR{\string\sub@on}%
597   \else      \write\linenum@outR{\string\sub@on}%
598   \fi
599   \ifdim\dimen0>0pt \hskip\dimen0 \fi}
600 \def\endsub{\dimen0\lastskip
601   \ifdim\dimen0>0pt \unskip \fi}
```

```

602 \ifledRcol \write\linenum@outR{\string\sub@off}%
603 \else      \write\linenum@out{\string\sub@off}%
604 \fi
605 \ifdim\dimen0>0pt \hskip\dimen0 \fi}
606

```

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

```

607 \renewcommand*{\advanceline}[1]{%
608 \ifledRcol \write\linenum@outR{\string\@adv[#1]}%
609 \else      \write\linenum@out{\string\@adv[#1]}%
610 \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.

```

611 \renewcommand*{\setline}[1]{%
612 \ifnum#1<\z@
613 \led@warn@BadSetline
614 \else
615 \ifledRcol \write\linenum@outR{\string\@set[#1]}%
616 \else      \write\linenum@out{\string\@set[#1]}%
617 \fi
618 \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.

```

619 \renewcommand*{\setlinenum}[1]{%
620 \ifnum#1<\z@
621 \led@warn@BadSetlinenum
622 \else
623 \ifledRcol \write\linenum@outR{\string\l@d@set[#1]}
624 \else      \write\linenum@out{\string\l@d@set[#1]} \fi
625 \fi}
626

```

\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.

```

627 \renewcommand*{\startlock}{%
628 \ifledRcol \write\linenum@outR{\string\lock@on}%
629 \else      \write\linenum@out{\string\lock@on}%
630 \fi}
631 \def\endlock{%
632 \ifledRcol \write\linenum@outR{\string\lock@off}%
633 \else      \write\linenum@out{\string\lock@off}%
634 \fi}
635

```

`\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.

```

636 \renewcommand*{\skipnumbering}{%
637   \ifledRcol \write\linenum@outR{\string\n@num}%
638             \advanceline{-1}%
639   \else
640     \skipnumbering@reg
641   \fi}
642

```

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.

```

643 \long\def\critext#1#2/{\leavevmode
644   \begingroup
645     \renewcommand{\@tag}{\no@expands #1}%
646     \set@line
647     \ifledRcol \global\insert@countR \z@
648     \else      \global\insert@count \z@ \fi
649     \ignorespaces #2\relax
650     \flag@start
651   \endgroup
652   \showlemma{#1}%
653   \ifx@end@lemmas\empty \else
654     \gl@p@end@lemmas\to\x@lemma
655     \x@lemma
656     \global\let\x@lemma=\relax
657   \fi
658   \flag@end}

```

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

```

659 \renewcommand{\edtext}[2]{\leavevmode
660   \begin{group}
661     \renewcommand{\@tag}{\no@expands #1}%
662     \set@line
663     \ifl@edRcol \global\insert@countR \z@
664     \else      \global\insert@count \z@ \fi
665     \ignorespaces #2\relax
666     \flag@start
667   \end{group}
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   \flag@end}
675
```

`\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`.

```

676 \renewcommand*{\set@line}{%
677   \ifl@edRcol
678     \ifx\line@listR\empty
679       \global\noteschanged@true
680       \xdef\l@d@nums{000|000|000|000|000|000|\edfont@info}%
681     \else
682       \gl@p\line@listR\to\@tempb
683       \xdef\l@d@nums{\@tempb|\edfont@info}%
684       \global\let\@tempb=\undefined
685     \fi
686   \else
687     \ifx\line@list\empty
688       \global\noteschanged@true
689       \xdef\l@d@nums{000|000|000|000|000|000|\edfont@info}%
690     \else
691       \gl@p\line@list\to\@tempb
692       \xdef\l@d@nums{\@tempb|\edfont@info}%
693       \global\let\@tempb=\undefined
694     \fi
695   \fi}
696
```

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
`pages`
`chapterinpages`

parallel pages.

```
697 \newenvironment{pairs}{%}
698   \l@dpairingtrue
699   \l@dpagingfalse
700 }{%
701   \l@dpairingfalse
702 }
```

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.

```
703 \newenvironment{pages}{%
704   \let\oldchapter\chapter
705   \let\chapter\chapterinpages
706   \l@dpairingtrue
707   \l@dpagingtrue
708   \setlength{\Lcolwidth}{\textwidth}%
709   \setlength{\Rcolwidth}{\textwidth}%
710 }{%
711   \l@dpairingfalse
712   \l@dpagingfalse
713   \let\chapter\oldchapter
714 }
715 \newcommand{\chapterinpages}{\thispagestyle{plain}%
716                               \global\@topnum\z@
717                               \@afterindentfalse
718                               \secdef\@chapter\@schapter}
719
```

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

```
720 \newif\ifinstanzaL
721 \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.

```
722 \newenvironment{Leftside}{%
723   \ledRcolfalse
724   \let\beginnumbering\beginnumbering\setcounter{pstartL}{1}
725   \let\pstart\pstartL
726   \let\thepstart\thepstartL
727   \let\pend\pendL
728   \let\memorydump\memorydumpL
729   \Leftsidehook
730   \let\oldstanza\stanza
731   \renewcommand{\stanza}{\oldstanza\global\instanzaLtrue}
```

```

732 }{
733   \let\stanza\oldstanza
734   \Leftsidehookend}

\Leftsidehook Hooks into the start and end of the Leftside and Rightside environments. These
\Leftsidehookend are initially empty.
\Rightsidehook 735 \newcommand*{\Leftsidehook}{}
\Rightsidehookend 736 \newcommand*{\Leftsidehookend}{}
737 \newcommand*{\Rightsidehook}{}
738 \newcommand*{\Rightsidehookend}{}
739

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.

740 \newenvironment{Rightside}{%
741   \ledRcoltrue
742   \let\beginnumbering\beginnumberingR
743   \let\endnumbering\endnumberingR
744   \let\pausenumbering\pausenumberingR
745   \let\resumenumbering\resumenumberingR
746   \let\memorydump\memorydumpR
747   \let\thepstart\thepstartR
748   \let\pstart\pstartR
749   \let\pend\pendR
750   \let\lineation\lineationR
751   \Rightsidehook
752   \let\oldstanza\stanza
753   \renewcommand{\stanza}{\oldstanza\global\instanzaRtrue}
754 }{%
755   \ledRcolfalse
756   \let\stanza\oldstanza
757   \Rightsidehookend
758 }
759

```

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` 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.

```
760 \newcount\num@linesR
```

```
761 \newbox\one@lineR
```

```
762 \newcount\par@lineR
```

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

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

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

```
763
```

```
764 \newcounter{pstartL}
```

```
765 \newcounter{pstartLold}
```

```
766 \renewcommand{\thepstartL}{\bfseries\@arabic{c@pstartL}. }
```

```
767 \newcounter{pstartR}
```

```
768 \newcounter{pstartRold}
```

```
769 \renewcommand{\thepstartR}{\bfseries\@arabic{c@pstartR}. }
```

```
770
```

```
771 \newcommand*{\pstartL}{
```

```
772 \if@nobreak
```

```
773 \let\@oldnobreak\@nobreaktrue
```

```
774 \else
```

```
775 \let\@oldnobreak\@nobreakfalse
```

```
776 \fi
```

```
777 \@nobreaktrue
```

```
778 \ifnumbering \else
```

```
779 \led@err@PstartNotNumbered
```

```
780 \beginnumbering
```

```
781 \fi
```

```
782 \ifnumberedpar@
```

```
783 \led@err@PstartInPstart
```

```
784 \pend
```

785 \fi

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

```
786 \ifpst@rtedL\else
787   \setcounter{pstartLold}{\value{pstartL}}%
788   \list@clear{\inserts@list}%
789   \global\let\next@insert=\empty
790   \global\pst@rtedLtrue
791 \fi
792 \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.

```
793 \global\advance\l@dnumpstartsL \@ne
794 \ifnum\l@dnumpstartsL>\l@dc@maxchunks
795   \led@err@TooManyPstarts
796   \global\l@dnumpstartsL=\l@dc@maxchunks
797 \fi
798 \global\setnamebox{\l@dLcolrawbox\the\l@dnumpstartsL}=\vbox\bgroup\ifautopar\else\ifnumb
799   \hsize=\Lcolwidth
800 \numberedpar@true
801 \iflabelpstart\protected@edef\@currentlabel
802   {\p@pstartL\thepstartL}\fi
803 }

804 \newcommand*{\pstartR}{
805 \if@nobreak
806   \let\@oldnobreak\@nobreaktrue
807 \else
808   \let\@oldnobreak\@nobreakfalse
809 \fi
810   \@nobreaktrue
811 \ifnumberingR \else
812   \led@err@PstartNotNumbered
813   \beginnumberingR
814 \fi
815 \ifnumberedpar@
816   \led@err@PstartInPstart
817   \pendR
818 \fi
819 \ifpst@rtedR\else
820   \setcounter{pstartRold}{\value{pstartR}}%
821   \list@clear{\inserts@listR}%
822   \global\let\next@insertR=\empty
823   \global\pst@rtedRtrue
824 \fi
825 \begingroup\normal@pars
826 \global\advance\l@dnumpstartsR \@ne
827 \ifnum\l@dnumpstartsR>\l@dc@maxchunks
828   \led@err@TooManyPstarts
```

```

829   \global\l@dnumpstartsR=\l@dc@maxchunks
830   \fi
831   \global\setnamebox{l@dRcolrawbox\the\l@dnumpstartsR}=\vbox\bgroup\ifautopar\else\ifnumberpstart\ifs
832   \hsize=\Rcolwidth
833   \numberedpar@true
834   \iflabelpstart\protected@edef\@currentlabel
835   {\p@pstartR\thepstartR}
836   }

```

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

```

837 \newcommand*{\pendL}{\ifnumbering \else
838   \led@err@PendNotNumbered
839   \fi
840   \ifnumberedpar@ \else
841   \led@err@PendNoPstart
842   \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.

```

843   \l@dzeropenalties
844   \endgraf\global\num@lines=\prevgraf\egroup
845   \global\par@line=0

```

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

```

846   \endgroup
847   \ignorespaces
848   \@oldnbreak
849   \ifnumberpstart
850   \addtocounter{pstartL}{1}
851   \fi}
852

```

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

```

853 \newcommand*{\pendR}{\ifnumberingR \else
854   \led@err@PendNotNumbered
855   \fi
856   \ifnumberedpar@ \else
857   \led@err@PendNoPstart
858   \fi
859   \l@dzeropenalties
860   \endgraf\global\num@linesR=\prevgraf\egroup
861   \global\par@lineR=0
862   \endgroup
863   \ignorespaces
864   \@oldnbreak
865   \ifnumberpstart

```

```

866 \addtocounter{pstartR}{1}
867 \fi
868 }
869

```

14.2 Processing one line

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

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

```

870 \newbox\l@dleftbox
871 \newbox\l@drightbox
872

```

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

```

\countRline 873 \newcount\countLline
874 \countLline \z@
875 \newcount\countRline
876 \countRline \z@
877

```

`\@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 878 \newcount\@donereallinesL
879 \newcount\@donetotallinesL
880 \newcount\@donereallinesR
881 \newcount\@donetotallinesR
882

```

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

```

883 \newcommand*{\do@lineL}{%
884 \advance\countLline \@one
885 \ifvbox\namebox{\l@dLcolrawbox\the\l@dpscl}%
886 {\vbadness=10000
887 \splittopskip=\z@
888 \do@lineLhook
889 \l@demtyd@ta
890 \global\setbox\one@line=\vsplit\namebox{\l@dLcolrawbox\the\l@dpscl}
891 to\baselineskip}%
892 \unvbox\one@line \global\setbox\one@line=\lastbox
893 \getline@numL
894 \ifnum\@lock>\@one\inserthangingsymboltrue\else\inserthangingsymbolfalse\fi
895 \setbox\l@dleftbox
896 \hb@xt@ \l@colwidth{%

```

```

897     \affixpstart@numL
898     \affixline@num
899     \l@dld@ta
900     \add@inserts
901     \affixside@note
902     \l@dlsn@te
903     {\ledllfill\hb@xt@ \wd\one@line{\inserthangingsymbolL\new@line\l@dunhbox@line{\one@line}}\correct
904     \l@drsn@te
905     }%
906     \add@penaltiesL
907     \global\advance\@donereallinesL\@ne
908     \global\advance\@donetotallinesL\@ne
909 \else
910     \setbox\l@dleftbox \hb@xt@ \Lcolwidth{\hspace*\Lcolwidth}%
911     \global\advance\@donetotallinesL\@ne
912 \fi}
913
914

```

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

```

\do@lineRhook 915 \newcommand*{\do@lineLhook}{}
916 \newcommand*{\do@lineRhook}{}
917

```

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

```

918 \newcommand*{\do@lineR}{%
919   \advance\countRline \@ne
920   \ifvbox\namebox{\l@dRcolrawbox\the\l@dpscR}%
921   {\vbadness=10000
922     \splittopskip=\z@
923     \do@lineRhook
924     \l@emptyd@ta
925     \global\setbox\one@lineR=\vsplit\namebox{\l@dRcolrawbox\the\l@dpscR}
926     to\baselineskip}%
927   \unvbox\one@lineR \global\setbox\one@lineR=\lastbox
928   \getline@numR
929   \ifnum\@lockR>\@ne\inserthangingsymbolRtrue\else\inserthangingsymbolRfalse\fi
930   \setbox\l@drightbox
931   \hb@xt@ \Rcolwidth{%
932     \affixpstart@numR
933     \affixline@numR
934     \l@dld@ta
935     \add@insertsR
936     \affixside@noteR
937     \l@dlsn@te
938     {\correcthangingsymbolR\ledllfill\hb@xt@ \wd\one@lineR{\inserthangingsymbolR\new@line\l@dunhbox@line{\
939     \l@drsn@te
940     }%
941     \add@penaltiesR

```

```

942 \global\advance\@donereallinesR\@ne
943 \global\advance\@donetotallinesR\@ne
944 \else
945 \setbox\l@drightbox \hb@xt@ \Rcolwidth{\hspace*{\Rcolwidth}}
946 \global\advance\@donetotallinesR\@ne
947 \fi}
948
949

```

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.

```

950 \newcommand*{\getline@numR}{%
951 \global\advance\absline@numR \@ne
952 \do@actionsR
953 \do@ballastR
954 \ifnumberline
955 \ifsublines@
956 \ifnum\sub@lockR<\tw@
957 \global\advance\subline@numR \@ne
958 \fi
959 \else
960 \ifnum\@lockR<\tw@
961 \global\advance\line@numR \@ne
962 \global\subline@numR \z@
963 \fi
964 \fi
965 \fi
966 }
967 \newcommand*{\getline@numL}{%
968 \global\advance\absline@num \@ne
969 \do@actions
970 \do@ballast
971 \ifnumberline
972 \ifsublines@
973 \ifnum\sub@lock<\tw@
974 \global\advance\subline@num \@ne
975 \fi
976 \else
977 \ifnum\@lock<\tw@
978 \global\advance\line@num \@ne
979 \global\subline@num \z@
980 \fi
981 \fi
982 \fi
983 }
984
985

```


`\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.

```

986 \newcommand*{\do@ballastR}{\global\ballast@count=\z@
987   \begingroup
988     \advance\absline@numR \@ne
989     \ifnum\next@actionlineR=\absline@numR
990       \ifnum\next@actionR>-1001
991         \global\advance\ballast@count by -\c@ballast
992       \fi
993     \fi
994   \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.

```

995 \newcommand*{\do@actions@fixedcodeR}{%
996   \ifcase\@l@tempcnta%
997     \or%                % 1001
998     \global\sublines@true
999     \or%                % 1002
1000    \global\sublines@false
1001    \or%                % 1003
1002    \global\@lockR=\@ne
1003    \or%                % 1004
1004    \ifnum\@lockR=\tw@
1005      \global\@lockR=\thr@@
1006    \else
1007      \global\@lockR=\z@
1008    \fi
1009    \or%                % 1005
1010    \global\sub@lockR=\@ne
1011    \or%                % 1006
1012    \ifnum\sub@lockR=\tw@
1013      \global\sub@lockR=\thr@@
1014    \else
1015      \global\sub@lockR=\z@
1016    \fi
1017    \or%                % 1007
1018    \l@dskipnumbertrue
1019  \else
1020    \led@warn@BadAction
1021  \fi}
1022
1023
1024 \newcommand*{\do@actionsR}{%
1025   \global\let\do@actions@nextR=\relax
1026   \@l@tempcntb=\absline@numR

```

```

1027 \ifnum\@l@tempcntb<\next@actionlineR\else
1028 \ifnum\next@actionR>-1001\relax
1029 \global\page@numR=\next@actionR
1030 \ifbypage@R
1031 \global\line@numR \z@ \global\subline@numR \z@
1032 \fi
1033 \else
1034 \ifnum\next@actionR<-4999\relax % 9/05 added relax here
1035 \@l@tempcnta=-\next@actionR
1036 \advance\@l@tempcnta by -5001\relax
1037 \ifsublines@
1038 \global\subline@numR=\@l@tempcnta
1039 \else
1040 \global\line@numR=\@l@tempcnta
1041 \fi
1042 \else
1043 \@l@tempcnta=-\next@actionR
1044 \advance\@l@tempcnta by -1000\relax
1045 \do@actions@fixedcodeR
1046 \fi
1047 \fi
1048 \ifx\actionlines@listR\empty
1049 \gdef\next@actionlineR{1000000}%
1050 \else
1051 \gl@p\actionlines@listR\to\next@actionlineR
1052 \gl@p\actions@listR\to\next@actionR
1053 \global\let\do@actions@nextR=\do@actionsR
1054 \fi
1055 \fi
1056 \do@actions@nextR}
1057

```

14.4 Line number printing

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

```

\ch@cksub@l@ckR 1058
\ch@ck@l@ckR 1059 \providecommand*\l@dcalcnum}[3]{%
\fx@l@cksR 1060 \ifnum #1 > #2\relax
\affixline@numR 1061 \@l@tempcnta = #1\relax
1062 \advance\@l@tempcnta by -#2\relax
1063 \divide\@l@tempcnta by #3\relax
1064 \multiply\@l@tempcnta by #3\relax
1065 \advance\@l@tempcnta by #2\relax
1066 \else
1067 \@l@tempcnta=#2\relax
1068 \fi}
1069
1070 \newcommand*\ch@cksub@l@ckR}{%
1071 \ifcase\sub@lockR

```

```

1072 \or
1073   \ifnum\sublock@disp=\@ne
1074     \@l@tempcntb \z@ \@l@tempcnta \@ne
1075   \fi
1076 \or
1077   \ifnum\sublock@disp=\tw@
1078   \else
1079     \@l@tempcntb \z@ \@l@tempcnta \@ne
1080   \fi
1081 \or
1082   \ifnum\sublock@disp=\z@
1083     \@l@tempcntb \z@ \@l@tempcnta \@ne
1084   \fi
1085 \fi}
1086
1087 \newcommand*{\ch@ck@l@ckR}{%
1088   \ifcase\@lockR
1089   \or
1090     \ifnum\lock@disp=\@ne
1091       \@l@tempcntb \z@ \@l@tempcnta \@ne
1092     \fi
1093   \or
1094     \ifnum\lock@disp=\tw@
1095     \else
1096       \@l@tempcntb \z@ \@l@tempcnta \@ne
1097     \fi
1098   \or
1099     \ifnum\lock@disp=\z@
1100       \@l@tempcntb \z@ \@l@tempcnta \@ne
1101     \fi
1102   \fi}
1103
1104 \newcommand*{\f@x@l@cksR}{%
1105   \ifcase\@lockR
1106   \or
1107     \global\@lockR \tw@
1108   \or \or
1109     \global\@lockR \z@
1110   \fi
1111   \ifcase\sub@lockR
1112   \or
1113     \global\sub@lockR \tw@
1114   \or \or
1115     \global\sub@lockR \z@
1116   \fi}
1117
1118
1119 \newcommand*{\affixline@numR}{%
1120 \ifnumberline
1121 \ifl@dskipnumber

```

```

1122 \global\l@dskipnumberfalse
1123 \else
1124 \ifsublines@
1125 \l@dttempcntb=\subline@numR
1126 \l@dcalcnnum{\subline@numR}{\c@firstsublinenumR}{\c@sublinenumincrementR}%
1127 \ch@cksub@lockR
1128 \else
1129 \l@dttempcntb=\line@numR
1130 \ifx\linenumberlist\empty
1131 \l@dcalcnnum{\line@numR}{\c@firstlinenumR}{\c@linenumincrementR}%
1132 \else
1133 \l@dttempcnta=\line@numR
1134 \edef\rem@inder{\linenumberlist,\number\line@numR,}%
1135 \edef\sc@n@list{\def\noexpand\sc@n@list
1136 ###1,\number\l@dttempcnta,###2|{\def\noexpand\rem@inder{###2}}}%
1137 \sc@n@list\expandafter\sc@n@list\rem@inder|
1138 \ifx\rem@inder\empty\advance\l@dttempcnta\@ne\fi
1139 \fi
1140 \ch@ck@l@ckR
1141 \fi
1142 \ifnum\l@dttempcnta=\l@dttempcntb
1143 \if@twocolumn
1144 \if@firstcolumn
1145 \gdef\l@dld@ta{\llap{\leftlinenumR}}}%
1146 \else
1147 \gdef\l@drd@ta{\rlap{\rightlinenumR}}}%
1148 \fi
1149 \else
1150 \l@dttempcntb=\line@marginR
1151 \ifnum\l@dttempcntb>\@ne
1152 \advance\l@dttempcntb by\page@numR
1153 \fi
1154 \ifodd\l@dttempcntb
1155 \gdef\l@drd@ta{\rlap{\rightlinenumR}}}%
1156 \else
1157 \gdef\l@dld@ta{\llap{\leftlinenumR}}}%
1158 \fi
1159 \fi
1160 \fi
1161 \f@x@l@cksR
1162 \fi
1163 \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 1164
  \leftpstartnumR 1165 \newcommand*{\affixpstart@numL}{%
\rightpstartnumR 1166 \ifsidepstartnum
  \leftpstartnumL 1167 \if@twocolumn
\rightpstartnumL 1168   \if@firstcolumn
  \ifpstartnumR 1169     \gdef\l@dld@ta{\llap{\leftpstartnumL}}}%
1170     \else
1171     \gdef\l@drd@ta{\rlap{\rightpstartnumL}}}%
1172     \fi
1173   \else
1174     \l@dtempcntb=\line@margin
1175     \ifnum\l@dtempcntb>\@ne
1176       \advance\l@dtempcntb \page@num
1177     \fi
1178     \ifodd\l@dtempcntb
1179       \gdef\l@drd@ta{\rlap{\rightpstartnumL}}}%
1180     \else
1181       \gdef\l@dld@ta{\llap{\leftpstartnumL}}}%
1182     \fi
1183   \fi
1184 \fi
1185 }
1186 \newcommand*{\affixpstart@numR}{%
1187 \ifsidepstartnum
1188 \if@twocolumn
1189   \if@firstcolumn
1190     \gdef\l@dld@ta{\llap{\leftpstartnumR}}}%
1191   \else
1192     \gdef\l@drd@ta{\rlap{\rightpstartnumR}}}%
1193   \fi
1194   \else
1195     \l@dtempcntb=\line@marginR
1196     \ifnum\l@dtempcntb>\@ne
1197       \advance\l@dtempcntb \page@numR
1198     \fi
1199     \ifodd\l@dtempcntb
1200       \gdef\l@drd@ta{\rlap{\rightpstartnumR}}}%
1201     \else
1202       \gdef\l@dld@ta{\llap{\leftpstartnumR}}}%
1203     \fi
1204   \fi
1205 \fi
1206 }
1207
1208 \newcommand*{\leftpstartnumL}{
1209 \ifpstartnum

```

```

1210 \thepstartL
1211 \kern\linenumsep\global\pstartnumfalse\fi
1212 }
1213 \newcommand*{\rightpstartnumL}{
1214 \ifpstartnum\kern\linenumsep
1215 \thepstartL
1216 \global\pstartnumfalse\fi
1217 }
1218 \newif\ifpstartnumR
1219 \pstartnumRtrue
1220 \newcommand*{\leftpstartnumR}{
1221 \ifpstartnumR
1222 \thepstartR
1223 \kern\linenumsep\global\pstartnumRfalse\fi
1224 }
1225 \newcommand*{\rightpstartnumR}{
1226 \ifpstartnumR\kern\linenumsep
1227 \thepstartR
1228 \global\pstartnumRfalse\fi
1229 }

```

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.

```

1230 \list@create{\inserts@listR}

```

`\add@insertsR` The right text version.

```

\add@inserts@nextR 1231 \newcommand*{\add@insertsR}{%
1232 \global\let\add@inserts@nextR=\relax
1233 \ifx\inserts@listR\empty \else
1234 \ifx\next@insertR\empty
1235 \ifx\insertlines@listR\empty
1236 \global\noteschanged@true
1237 \gdef\next@insertR{100000}%
1238 \else
1239 \gl@p\insertlines@listR\to\next@insertR
1240 \fi
1241 \fi
1242 \ifnum\next@insertR=\absline@numR
1243 \gl@p\inserts@listR\to\@insertR
1244 \@insertR
1245 \global\let\@insertR=\undefined
1246 \global\let\next@insertR=\empty
1247 \global\let\add@inserts@nextR=\add@insertsR
1248 \fi
1249 \fi
1250 \add@inserts@nextR}
1251

```

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.

```
1252 \newcommand*{\add@penaltiesL}{ }
1253 \newcommand*{\add@penaltiesR}{ }
1254
```

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.

```
1255 \newcommand*{\flush@notesR}{%
1256   \@xloop
1257   \ifx\inserts@listR\empty \else
1258     \gl@p\inserts@listR\to\@insertR
1259     \@insertR
1260     \global\let\@insertR=\undefined
1261   \repeat}
1262
```

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
```

```
1263 \def\printlinesR#1|#2|#3|#4|#5|#6|#7|{\begingroup
1264   \setprintlines{#1}{#2}{#3}{#4}{#5}{#6}%
1265   \ifl@d@pnum #1\fullstop\fi
1266   \ifledplinenum \linenumr@p{#2}\Rlineflag\else \symplinenum\fi
1267   \ifl@d@ssub \fullstop \sublinenumr@p{#3}\fi
1268   \ifl@d@dash \endashchar\fi
1269   \ifl@d@pnum #4\fullstop\fi
1270   \ifl@d@elin \linenumr@p{#5}\Rlineflag\fi
1271   \ifl@d@esl \ifl@d@elin \fullstop\fi \sublinenumr@p{#6}\fi
1272 \endgroup}
1273
1274 \let\ledsavedprintlines\printlines
1275
```

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.

```
1276 \list@create{\labelref@listR}
```


1277

`\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.

```

1278 \renewcommand*{\edlabel}[1]{\@bsphack
1279   \ifledRcol
1280     \write\linenum@outR{\string\@lab}%
1281     \ifx\labelref@listR\empty
1282       \xdef\label@refs{\zz@@@}%
1283     \else
1284       \glp\labelref@listR\to\label@refs
1285     \fi
1286     \ifvmode
1287       \advancelabel@refs
1288     \fi
1289     \protected@write\@auxout{%
1290       {\string\l@dmake@labelsR\space\thepage|\label@refs|{#1}}%
1291   \else
1292     \write\linenum@out{\string\@lab}%
1293     \ifx\labelref@list\empty
1294       \xdef\label@refs{\zz@@@}%
1295     \else
1296       \glp\labelref@list\to\label@refs
1297     \fi
1298     \ifvmode
1299       \advancelabel@refs
1300     \fi
1301     \protected@write\@auxout{%
1302       {\string\l@dmake@labelsR\space\thepage|\label@refs|{#1}}%
1303   \fi
1304   \@esphack}
1305
```

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

```

1306 \def\l@dmake@labelsR#1|#2|#3|#4{%
1307   \expandafter\ifx\csname the@label#4\endcsname \relax\else
1308     \led@warn@DuplicateLabel{#4}%
1309   \fi
1310   \expandafter\gdef\csname the@label#4\endcsname{#1|#2\Rlineflag|#3}%
1311   \ignorespaces}
1312 \AtBeginDocument{%
1313   \def\l@dmake@labelsR#1|#2|#3|#4{%
1314   }
1315
```

`\@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.

```

1316 \renewcommand*{\@lab}{%
1317   \ifledRcol
1318     \xright@appenditem{\linenumr@p{\line@numR}}{|%
1319       \ifsublines@ \sublinenumr@p{\subline@numR}\else 0\fi}%
1320     \to\labelref@listR
1321   \else
1322     \xright@appenditem{\linenumr@p{\line@num}}{|%
1323       \ifsublines@ \sublinenumr@p{\subline@num}\else 0\fi}%
1324     \to\labelref@list
1325   \fi}
1326
```

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 1327 \newcount\sidenote@marginR
1328 \renewcommand*{\sidenotemargin}[1]{%
1329   \l@dgetsidenote@margin{#1}%
1330   \ifnum\@l@dttempcntb>\m@ne
1331     \ifledRcol
1332       \global\sidenote@marginR=\@l@dttempcntb
1333     \else
1334       \global\sidenote@margin=\@l@dttempcntb
1335     \fi
1336   \fi}}
1337 \sidenotemargin{right}
1338 \global\sidenote@margin=\@ne
1339
```

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

```

\l@dcsnote 1340 \renewcommand*{\l@dlsnote}[1]{%
1341   \ifnumberedpar@
1342     \ifledRcol%
1343       \xright@appenditem{\noexpand\l@dlsnote{#1}}{%
1344         \to\inserts@listR
1345       \else%
1346       \xright@appenditem{\noexpand\l@dlsnote{#1}}{%
1347         \to\inserts@list
1348       \global\advance\insert@count \@ne%
1349       \fi
1350     \fi\ignorespaces}
1351 \renewcommand*{\l@drsnote}[1]{%

```

```

1352 \ifnumberedpar@
1353 \ifledRcol%
1354 \xright@appenditem{\noexpand\vl@drsnote{#1}}%
1355 \to\inserts@listR
1356 \global\advance\insert@countR \@ne%
1357 \else%
1358 \xright@appenditem{\noexpand\vl@drsnote{#1}}%
1359 \to\inserts@list
1360 \global\advance\insert@count \@ne%
1361 \fi
1362 \fi\ignorespaces}
1363 \renewcommand*{\l@dcnote}[1]{%
1364 \ifnumberedpar@
1365 \ifledRcol%
1366 \xright@appenditem{\noexpand\vl@dcnote{#1}}%
1367 \to\inserts@listR
1368 \global\advance\insert@countR \@ne%
1369 \else%
1370 \xright@appenditem{\noexpand\vl@dcnote{#1}}%
1371 \to\inserts@list
1372 \global\advance\insert@count \@ne%
1373 \fi
1374 \fi\ignorespaces}
1375

```

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

```

1376 \newcommand*{\affixside@noteR}{%
1377 \def\sidenotecontent@{%
1378 \numdef{\itemcount@}{0}%
1379 \renewcommand{\do}[1]{%
1380 \ifnumequal{\itemcount@}{0}%
1381 {%
1382 \appto\sidenotecontent@{##1}}% Not print not separator before the 1st note
1383 {\appto\sidenotecontent@{\sidenotesep ##1}}%
1384 }%
1385 \numdef{\itemcount@}{\itemcount@+1}%
1386 }%
1387 \dolistloop{\l@dcnotetext}%
1388 \ifnumgreater{\itemcount@}{1}{\eledmac@warning{\itemcount@\space sidenotes on line \the\line@numR}}%
1389 \gdef\@templ@d{%
1390 \ifx\@templ@d\l@dcnotetext \else
1391 \if@twocolumn
1392 \if@firstcolumn
1393 \setl@dlp@rbox{\sidenotecontent@}%
1394 \else
1395 \setl@drp@rbox{\sidenotecontent@}%
1396 \fi
1397 \else
1398 \l@dttempcntb=\sidenote@marginR
1399 \ifnum\l@dttempcntb>\@ne

```

```

1400      \advance\@l@dttempcntb by\page@num
1401      \fi
1402      \ifodd\@l@dttempcntb
1403        \setl@drp@rbox{\sidenotecontent@t}%
1404      \else
1405        \setl@dlp@rbox{\sidenotecontent@}%
1406      \fi
1407    \fi
1408  \fi}
1409

```

18 Familiar footnotes

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

```

1410 \renewcommand{\l@dbfnote}[1]{%
1411   \ifnumberedpar@
1412     \ifledRcol%
1413       \xright@appenditem{\noexpand\vl@dbfnote{#1}{\@thefnmark}}%
1414                           \to\inserts@listR
1415       \global\advance\insert@countR \@ne%
1416     \else%
1417       \xright@appenditem{\noexpand\vl@dbfnote{#1}{\@thefnmark}}%
1418                           \to\inserts@list
1419       \global\advance\insert@count \@ne%
1420     \fi
1421   \fi\ignorespaces}
1422

```

`\normalbfnoteX`

```

1423 \renewcommand{\normalbfnoteX}[2]{%
1424   \ifnumberedpar@
1425     \ifledRcol%
1426       \xright@appenditem{\noexpand\vbfnoteX{#1}{#2}{\@nameuse{thefootnote#1}}}%
1427                           \to\inserts@listR
1428       \global\advance\insert@countR \@ne%
1429     \else%
1430       \xright@appenditem{\noexpand\vbfnoteX{#1}{#2}{\@nameuse{thefootnote#1}}}%
1431                           \to\inserts@list
1432       \global\advance\insert@count \@ne%
1433     \fi
1434   \fi\ignorespaces}
1435

```

19 Verse

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

```

\inserthangingsymbolL
\inserthangingsymbolR 1436 \newif\ifinserthangingsymbolR
                        1437 \newcommand{\inserthangingsymbolL}{%
                        1438 \ifinserthangingsymbol%
                        1439     \ifinstanzaL%
                        1440         \hfill\hangingsymbol%
                        1441     \fi%
                        1442 \fi}
                        1443 \newcommand{\inserthangingsymbolR}{%
                        1444 \ifinserthangingsymbolR%
                        1445     \ifinstanzaR%
                        1446         \hfill\hangingsymbol%
                        1447     \fi%
                        1448 \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 1449 \newcommand{\correcthangingL}{%
                  1450 \ifl@dpadding\else%
                  1451     \ifinstanzaL%
                  1452         \ifinserthangingsymbol%
                  1453             \hskip \@ifundefined{sza@00}{0}{\expandafter%
                  1454                 \noexpand\csname sza@00\endcsname}\stanzaindentbase%
                  1455         \fi%
                  1456     \fi%
                  1457 \fi}
                  1458
                  1459 \newcommand{\correcthangingR}{%
                  1460 \ifl@dpadding\else%
                  1461     \ifinstanzaR%
                  1462         \ifinserthangingsymbolR%
                  1463             \hskip \@ifundefined{sza@00}{0}{\expandafter%
                  1464                 \noexpand\csname sza@00\endcsname}\stanzaindentbase%
                  1465         \fi%
                  1466     \fi%
                  1467 \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.

```

1468 \chardef\next=\catcode'\&
1469 \catcode'\&=\active
1470

```

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

```

1471 \newenvironment{astanza}{%
1472   \startstanzahook
1473   \catcode'\&\active
1474   \global\stanza@count\@ne
1475   \ifnum\usernamecount{sza@0@}=\z@
1476     \let\stanza@hang\relax
1477     \let\endlock\relax
1478   \else
1479   %%      \interlinepenalty\@M % this screws things up, but I don't know why
1480   \rightskip\z@ plus 1fil\relax
1481   \fi
1482   \ifnum\usernamecount{szp@0@}=\z@
1483     \let\sza@penalty\relax
1484   \fi
1485   \def&{%
1486     \endlock\mbox{}}%
1487     \sza@penalty
1488     \global\advance\stanza@count\@ne
1489     \@astanza@line}%
1490   \def\&{%
1491     \endlock\mbox{}}
1492     \pend
1493     \endstanzaextra}%
1494   \pstart
1495   \@astanza@line
1496 }{}
1497

```

`\@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.

```

1498 \newcommand*{\@astanza@line}{%
1499   \parindent=\csname sza@\number\stanza@count @\endcsname\stanzaindentbase
1500   \par
1501   \stanza@hang%\mbox{}}%
1502   \ignorespaces}
1503

```

Lastly reset the modified category codes.

```

1504   \catcode'\&=\next
1505

```

20 Naming macros

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

```

\newnamebox A set of macros for creating and using ‘named’boxes; the macros are called after
\setnamebox the regular box macros, but including the string ‘name’.
\unhnamebox
\unvnamebox
\namebox

```

```

1506 \providecommand*\newnamebox}[1]{%
1507   \expandafter\newbox\csname #1\endcsname}
1508 \providecommand*\setnamebox}[1]{%
1509   \expandafter\setbox\csname #1\endcsname}
1510 \providecommand*\unhnamebox}[1]{%
1511   \expandafter\unhbox\csname #1\endcsname}
1512 \providecommand*\unvnamebox}[1]{%
1513   \expandafter\unvbox\csname #1\endcsname}
1514 \providecommand*\namebox}[1]{%
1515   \csname #1\endcsname}
1516

```

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

```

\usernamecount 1517 \providecommand*\newnamecount}[1]{%
1518   \expandafter\newcount\csname #1\endcsname}
1519 \providecommand*\usernamecount}[1]{%
1520   \csname #1\endcsname}
1521

```

21 Counts and boxes for parallel texts

In sequential text, each chunk (that enclosed by `\pstart ... \pend`) is put into a box called `\raw@text` and then immediately printed, resulting in the box being emptied and ready for the next chunk. For parallel processing multiple boxes are needed as printing is delayed. We also need extra counters for various things.

`\maxchunks` The maximum number of chunk pairs before printing has to be called for. The default is 5120 chunk pairs.

```

1522 \newcount\l@dc@maxchunks
1523 \newcommand*\maxchunks}[1]{\l@dc@maxchunks=#1}
1524 \maxchunks{5120}
1525

```

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

```

\l@dnumpstartsR 1526 \newcount\l@dnumpstartsR
1527

```

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

```

\l@pscR 1528 \newcount\l@dpscL
1529 \newcount\l@dpscR
1530

```

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

```

1531 \newcommand*\l@dsetuprawboxes){%
1532   \@l@dttempcntb=\l@dc@maxchunks
1533   \loop\ifnum\@l@dttempcntb>\z@
1534     \newnamebox{\l@dLcolrawbox\the\@l@dttempcntb}

```

```

1535 \newnamebox{\l@Rcolrawbox\the\@l@tempcntb}
1536 \advance\@l@tempcntb \m@ne
1537 \repeat}
1538

```

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

```

1539 \newcommand*{\l@dssetupmaxlinecounts}{%
1540 \l@tempcntb=\l@dc@maxchunks
1541 \loop\ifnum\@l@tempcntb>\z@
1542 \newnamecount{\l@dsmaxlinesinpar\the\@l@tempcntb}
1543 \advance\@l@tempcntb \m@ne
1544 \repeat}
1545 \newcommand*{\l@dszeromaxlinecounts}{%
1546 \begingroup
1547 \l@tempcntb=\l@dc@maxchunks
1548 \loop\ifnum\@l@tempcntb>\z@
1549 \global\usenamecount{\l@dsmaxlinesinpar\the\@l@tempcntb}=\z@
1550 \advance\@l@tempcntb \m@ne
1551 \repeat
1552 \endgroup}
1553

```

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

```

1554 \AtBeginDocument{%
1555 \l@dssetuprawboxes
1556 \l@dssetupmaxlinecounts
1557 \l@dszeromaxlinecounts
1558 \l@dsnumpstartsL=\z@
1559 \l@dsnumpstartsR=\z@
1560 \l@dspscL=\z@
1561 \l@dspscR=\z@}
1562

```

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 1563 \newif\ifl@dusedbabel
\l@dusedbabeltrue 1564 \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 1565 \newif\ifl@dsamelang
1566 \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.
1567 \newcommand*{\l@dchecklang}{%
1568 \l@dsamelangfalse
1569 \edef\@tempa{\theledlanguageL}\edef\@tempb{\theledlanguageR}%
1570 \ifx\@tempa\@tempb
1571 \l@dsamelangtrue
1572 \fi}
1573

\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 \.
1574 \newcommand*{\l@dbbl@set@language}[1]{%
1575 \edef\language{#1}%
1576 \select@language{\language}%
1577 \if@filesw
1578 \protected@write\auxout{\string\select@language{\language}}%
1579 \addtocontents{toc}{\string\select@language{\language}}%
1580 \addtocontents{lof}{\string\select@language{\language}}%
1581 \addtocontents{lot}{\string\select@language{\language}}%
1582 \fi}
1583

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 1584 \providecommand{\selectlanguage}[1]{%
1585 \newcommand*{\l@duselanguage}[1]{%
1586 \gdef\theledlanguageL{
1587 \gdef\theledlanguageR{
1588

```

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

```
1589 \AtBeginDocument{%
1590   \ifundefined{xpg@main@language}{%
1591     \ifundefined{bbl@main@language}{%
```

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

```
1592   \l@dusedbabelfalse
1593   \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.

```
1594   \l@dusedbabeltrue
1595   \let\l@doldselectlanguage\selectlanguage
1596   \let\l@doldbbl@set@language\bbl@set@language
1597   \let\bbl@set@language\l@doldbbl@set@language
1598   \renewcommand{\selectlanguage}[1]{%
1599     \l@doldselectlanguage{#1}%
1600     \ifledRcol \gdef\theledlanguageR{#1}%
1601     \else      \gdef\theledlanguageL{#1}%
1602     \fi}
```

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

```
1603   \renewcommand*{\l@duselanguage}[1]{%
1604     \l@doldselectlanguage{#1}}
```

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

```
1605   \gdef\theledlanguageL{\bbl@main@language}%
1606   \gdef\theledlanguageR{\bbl@main@language}%
1607   }%
1608 }
```

If on `Polyglossia`

```
1609 { \apptocmd{\xpg@set@language}{%
1610   \ifledRcol \gdef\theledlanguageR{#1}%
1611   \else      \gdef\theledlanguageL{#1}%
1612   \fi}%
1613   \let\l@duselanguage\xpg@set@language
1614   \gdef\theledlanguageL{\xpg@main@language}%
1615   \gdef\theledlanguageR{\xpg@main@language}%
1616 % \end{macrocode}
1617 % That's it.
1618 %   \begin{macrocode}
1619 }}
```

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.

```

1620 \newcommand*{\Columns}{%
1621   \setcounter{pstartL}{\value{pstartLold}}
1622   \setcounter{pstartR}{\value{pstartRold}}
1623   \ifnum\l@dnumstartsL=\l@dnumstartsR\else
1624     \led@err@BadLeftRightPstarts{\the\l@dnumstartsL}{\the\l@dnumstartsR}%
1625   \fi

```

Start a group and zero counters, etc.

```

1626 \begingroup
1627   \l@dzeropenalties
1628   \endgraf\global\num@lines=\prevgraf
1629   \global\num@linesR=\prevgraf
1630   \global\par@line=\z@
1631   \global\par@lineR=\z@
1632   \global\l@dpscL=\z@
1633   \global\l@dpscR=\z@

```

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

```

1634   \check@pstarts
1635   \loop\if@pstarts
1636     \global\pstartnumtrue
1637     \global\pstartnumRtrue

```

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

```

1638   \global\advance\l@dpscL \@ne
1639   \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

```

1640   \checkraw@text
1641   \l@dchecklang
1642 {   \loop\ifaraw@text

```

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

```

1643     \ifl@dsamelang
1644       \do@lineL
1645       \do@lineR
1646     \else
1647       \l@duselanguage{\theledlanguageL}%
1648       \do@lineL
1649       \l@duselanguage{\theledlanguageR}%
1650       \do@lineR
1651     \fi
1652     \hb@xt@ \hsize{%
1653       \hfill \unhbox\l@dleftbox
1654       \hfill \columnseparator \hfill
1655       \unhbox\l@drightbox

```

```

1656      }%
1657      \checkdraw@text
1658      \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.

```

1659      \@writelinesinparL
1660      \@writelinesinparR
1661      \check@pstarts
1662      \ifbypstart@
1663          \write\linenum@out{\string\@set[1]}
1664          \resetprevline@
1665      \fi
1666      \ifbypstart@R
1667          \write\linenum@outR{\string\@set[1]}
1668          \resetprevline@
1669      \fi
1670      \addtocounter{pstartL}{1}
1671      \addtocounter{pstartR}{1}
1672      \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.

```

1673      \flush@notes
1674      \flush@notesR
1675      \endgroup
1676      \global\l@dpscL=\z@
1677      \global\l@dpscR=\z@
1678      \global\l@dnumstartsl=\z@
1679      \global\l@dnumstartsr=\z@
1680      \ignorespaces
1681      \global\instanzaLfalse
1682      \global\instanzaRfalse}
1683

```

`\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).

```

1684 \newcommand*{\columnseparator}{%
1685   \smash{\rule[-0.2\baselineskip]{\columnrulewidth}{1.05\baselineskip}}}
1686 \newdimen\columnrulewidth
1687 \columnrulewidth=\z@
1688

```

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

```

\@pstartstrue 1689 \newif\if@pstarts
\@pstartsfalse 1690 \newcommand*{\check@pstarts}{%
\check@pstarts

```

```

1691 \@pstartsfalse
1692 \ifnum\l@dnumpstartsL>\l@dpscL
1693   \@pstartstrue
1694 \else
1695   \ifnum\l@dnumpstartsR>\l@dpscR
1696     \@pstartstrue
1697   \fi
1698 \fi
1699 }
1700

```

\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 1701 \newif\ifaraw@text
1702   \araw@textfalse
1703 \newcommand*{\checkraw@text}{%
1704   \araw@textfalse
1705   \ifvbox\namebox{1@dLcolrawbox\the\l@dpscL}
1706     \araw@texttrue
1707   \else
1708     \ifvbox\namebox{1@dRcolrawbox\the\l@dpscR}
1709       \araw@texttrue
1710     \fi
1711   \fi
1712 }
1713

```

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

```

1714 \newcommand*{\@writelinesinparL}{%
1715   \edef\next{%
1716     \write\linenum@out{\string\@pend[\the\@donereallinesL]}}%
1717   \next
1718   \global\@donereallinesL \z@}
1719 \newcommand*{\@writelinesinparR}{%
1720   \edef\next{%
1721     \write\linenum@outR{\string\@pendR[\the\@donereallinesR]}}%
1722   \next
1723   \global\@donereallinesR \z@}
1724

```

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

```

1725 \newcount\numpagelinesL
1726 \newcount\numpagelinesR
1727 \newcount\l@dminpagelines
1728

```

\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.

```

1729 \newcommand*{\Pages}{%
1730   \setcounter{pstartL}{\value{pstartLold}}
1731   \setcounter{pstartR}{\value{pstartRold}}
1732   \typeout{}
1733   \typeout{***** PAGES *****}
1734   \ifnum\l@dnumpstartsL=\l@dnumpstartsR\else
1735     \led@err@BadLeftRightPstarts{\the\l@dnumpstartsL}{\the\l@dnumpstartsR}%
1736   \fi

```

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

```

1737 \cleartol@evenpage
1738 \begingroup
1739   \l@dzeropenalties
1740   \endgraf\global\num@lines=\prevgraf
1741   \global\num@linesR=\prevgraf
1742   \global\par@line=\z@
1743   \global\par@lineR=\z@
1744   \global\l@dpscL=\z@
1745   \global\l@dpscR=\z@
1746   \writtenlinesLfalse
1747   \writtenlinesRfalse

```

Check if there are chunks to be processed.

```

1748   \check@pstarts
1749   \loop\if@pstarts

```

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

```

1750     \global\advance\l@dpscL \@ne
1751     \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**.

```

1752     \getlinesfromparlistL
1753     \getlinesfromparlistR
1754     \l@dcalc@maxoftwo{\cs@linesinparL}{\cs@linesinparR}%
1755     {\usernamecount{\l@dmaxlinesinpar\the\l@dpscL}}%
1756     \check@pstarts
1757   \repeat

```

Zero the counts again, ready for the next bit.

```

1758   \global\l@dpscL=\z@
1759   \global\l@dpscR=\z@

```

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

```
1760 \getlinesfrompagelistL
1761 \getlinesfrompagelistR
1762 \l@dcalc@minoftwo{\@cs@linesonpageL}{\@cs@linesonpageR}%
1763 {\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.

```
1764 \check@pstarts
1765 \if@pstarts
```

Increment the chunk counts to get the first pair.

```
1766 \global\advance\l@dpscL \@ne
1767 \global\advance\l@dpscR \@ne
```

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

```
1768 \global\@donereallinesL=\z@
1769 \global\@donetotallinesL=\z@
1770 \global\@donereallinesR=\z@
1771 \global\@donetotallinesR=\z@
```

Start a loop over the boxes (chunks).

```
1772 \checkraw@text
1773 % \begingroup
1774 { \loop\ifaraw@text
```

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

```
1775 \checkpageL
1776 \l@duselanguage{\theledlanguageL}%
1777 %%% \begingroup
1778 { \loop\ifl@dsamepage
1779
```

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

```
1780 \do@lineL
1781 \advance\l@numpagelinesL \@ne
1782 \ifshiftedpstarts
1783 \ifdim\ht\l@dleftbox>0pt\hb@xt@ \hsize{\ledstrutL\unhbox\l@dleftbox}\fi%
1784 \else
1785 \hb@xt@ \hsize{\ledstrutL\unhbox\l@dleftbox}%
1786 \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.

```
1787
1788 \get@nextboxL
1789 \checkpageL
1790 \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.

```

1791      \ifl@dpagfull
1792      \@writelinesonpageL{\the\numpagelinesL}%
1793      \else
1794      \@writelinesonpageL{1000}%
1795      \fi

```

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

```

1796      \numpagelinesL \z@
1797      \clearl@dleftpage }%

```

Now do the same for the right text.

```

1798      \checkpageR
1799      \l@duselanguage{\theledlanguageR}%
1800 {
1801      \loop\ifl@dsamepage
1802      \do@lineR
1803      \advance\numpagelinesR \@ne
1804      \ifshiftedpstarts
1805          \ifdim\ht\l@drightbox>0pt\hb@xt@ \hsize{\ledstrutR\unhbox\l@drightbox}%
1806      \else
1807          \hb@xt@ \hsize{\ledstrutR\unhbox\l@drightbox}%
1808      \fi
1809      \get@nextboxR
1810      \checkpageR
1811      \repeat
1812      \ifl@dpagfull
1813      \@writelinesonpageR{\the\numpagelinesR}%
1814      \else
1815      \@writelinesonpageR{1000}%
1816      \fi
1817      \numpagelinesR=\z@

```

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

```

1817      \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.

```

1818      \checkraw@text
1819      \ifaraw@text
1820      \getlinesfrompagelistL
1821      \getlinesfrompagelistR
1822      \l@dcalc@minoftwo{\@cs@linesonpageL}{\@cs@linesonpageR}%
1823      {\l@dminpagelines}%
1824      \fi
1825      \repeat}

```


We have now output the text from all the chunks.

```
1826 \fi
```

Make sure that there are no inserts hanging around.

```
1827 \flush@notes
```

```
1828 \flush@notesR
```

```
1829 \endgroup
```

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

```
1830 \global\l@dpscL=\z@
```

```
1831 \global\l@dpscR=\z@
```

```
1832 \global\l@dnumpestartsL=\z@
```

```
1833 \global\l@dnumpestartsR=\z@
```

```
1834 \global\instanzaLfalse
```

```
1835 \global\instanzaRfalse
```

```
1836 \ignorespaces}
```

```
1837
```

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

```
\ledstrutR 1838 \newcommand*{\ledstrutL}{\strut}
```

```
1839 \newcommand*{\ledstrutR}{\strut}
```

```
1840
```

`\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.

```
1841 \providecommand{\cleartoevenpage}[1][\@empty]{%
```

```
1842 \clearpage
```

```
1843 \ifodd\c@page\hbox{ }#1\clearpage\fi}
```

```
1844 \newcommand*{\cleartol@devenpage}{%
```

```
1845 \ifdim\pagetotal<\topskip% on an empty page
```

```
1846 \else
```

```
1847 \clearpage
```

```
1848 \fi
```

```
1849 \ifodd\c@page\hbox{ }\clearpage\fi}
```

```
1850 \newcommand*{\clearl@dleftpage}{%
```

```
1851 \clearpage
```

```
1852 \ifodd\c@page\else
```

```
1853 \led@err@LeftOnRightPage
```

```
1854 \hbox{ }%
```

```
1855 \cleardoublepage
```

```
1856 \fi}
```

```
1857 \newcommand*{\clearl@drighthouse}{%
```

```
1858 \clearpage
```

```
1859 \ifodd\c@page
```

```
1860 \led@err@RightOnLeftPage
```

```
1861 \hbox{ }%
```

```

1862 \cleartoevenpage
1863 \fi}
1864

```

`\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 1865 \newcommand*{\getlinesfromparlistL}{%
1866 \ifx\linesinpar@listL\empty
1867 \gdef\@cs@linesinparL{0}%
1868 \else
1869 \gl@p\linesinpar@listL\to\@cs@linesinparL
1870 \fi}
1871 \newcommand*{\getlinesfromparlistR}{%
1872 \ifx\linesinpar@listR\empty
1873 \gdef\@cs@linesinparR{0}%
1874 \else
1875 \gl@p\linesinpar@listR\to\@cs@linesinparR
1876 \fi}
1877

```

`\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 1878 \newcommand*{\getlinesfrompagelistL}{%
1879 \ifx\linesonpage@listL\empty
1880 \gdef\@cs@linesonpageL{1000}%
1881 \else
1882 \gl@p\linesonpage@listL\to\@cs@linesonpageL
1883 \fi}
1884 \newcommand*{\getlinesfrompagelistR}{%
1885 \ifx\linesonpage@listR\empty
1886 \gdef\@cs@linesonpageR{1000}%
1887 \else
1888 \gl@p\linesonpage@listR\to\@cs@linesonpageR
1889 \fi}
1890

```

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

```

1891 \newcommand*{\@writelinesonpageL}[1]{%
1892 \edef\next{\write\linenum@out{\string\@lopL{#1}}}%
1893 \next}
1894 \newcommand*{\@writelinesonpageR}[1]{%
1895 \edef\next{\write\linenum@outR{\string\@lopR{#1}}}%
1896 \next}
1897

```

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

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

```

1898 \newcommand*{\l@dcalc@maxoftwo}[3]{%
1899   \ifnum #2>#1\relax
1900     #3=#2\relax
1901   \else
1902     #3=#1\relax
1903   \fi}
1904 \newcommand*{\l@dcalc@minoftwo}[3]{%
1905   \ifnum #2<#1\relax
1906     #3=#2\relax
1907   \else
1908     #3=#1\relax
1909   \fi}
1910

```

`\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@dpagetruefalse` `\ifl@dsamepage` are set FALSE.

```

\checkpageL 1911 \newif\ifl@dsamepage
\checkpageR 1912   \l@dsamepagetrue
1913   \newif\ifl@dpagetrue
1914   \newcommand*{\checkpageL}{%
1915     \l@dpagetrue
1916     \l@dsamepagetrue
1917     \check@goal
1918     \ifdim\pagetotal<\ledthegoal
1919       \ifnum\umpagelinesL<\l@dmminpagelines
1920       \else
1921         \l@dsamepagefalse
1922         \l@dpagetruefalse
1923       \fi
1924     \else
1925       \l@dsamepagefalse
1926       \l@dpagetrue
1927     \fi}
1928 \newcommand*{\checkpageR}{%
1929   \l@dpagetrue
1930   \l@dsamepagetrue
1931   \check@goal
1932   \ifdim\pagetotal<\ledthegoal
1933     \ifnum\umpagelinesR<\l@dmminpagelines
1934     \else
1935       \l@dsamepagefalse
1936       \l@dpagetruefalse
1937     \fi

```

```

1938 \else
1939     \l@dsamepagefalse
1940     \l@dpagfulltrue
1941 \fi}
1942

\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.

1943 \newdimen\ledthegoal
1944 \ifshiftedpstarts
1945     \newcommand*{\goalfraction}{0.95}
1946 \else
1947     \newcommand*{\goalfraction}{0.9}
1948 \fi
1949
1950 \newcommand*{\check@goal}{%
1951     \ledthegoal=\goalfraction\pagegoal}
1952

\ifwrittenlinesL Booleans for whether line data has been written to the section file.
\ifwrittenlinesL 1953 \newif\ifwrittenlinesL
1954 \newif\ifwrittenlinesR
1955

\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.

1956 \newcommand*{\get@nextboxL}{%
1957     \ifvbox\namebox{\l@dLcolrawbox\the\l@dpscl}% box is not empty

        The current box is not empty; do nothing.

1958     \else%                                box is empty

        The box is empty; check if enough lines (real and blank) have been output.

1959     \ifnum\usenamecount{\l@dmaxlinesinpar\the\l@dpscl}>\@donetotallinesL
1960     \else

        Sufficient lines have been output.

1961     \ifwrittenlinesL
1962     \else

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

1963         \@writelinesinparL
1964         \writtenlinesLtrue
1965     \fi
1966     \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.

```

```

1967     \writtenlinesLfalse
1968     \ifbypstart@
1969         \ifnum\value{pstartL}<\value{pstartLold}
1970         \else
1971             \global\line@num=0
1972             \resetprevline@
1973         \fi
1974     \fi
1975     \addtocounter{pstartL}{1}
1976     \global\pstartnumtrue
1977     \l@dcalc@maxoftwo{\the\usernamecount{1@dmaxlinesinpar\the\l@dpscL}}%
1978                     {\the\@donetotallinesL}%
1979                     {\usernamecount{1@dmaxlinesinpar\the\l@dpscL}}%
1980     \global\@donetotallinesL \z@
1981     \global\advance\l@dpscL \@ne
1982 \fi
1983 \fi
1984 \fi}

1985 \newcommand*{\get@nextboxR}{%
1986     \ifvbox\namebox{1@dRcolrawbox\the\l@dpscR}% box is not empty
1987     \else% box is empty
1988         \ifnum\usernamecount{1@dmaxlinesinpar\the\l@dpscR}>\@donetotallinesR
1989         \else
1990             \ifwrittenlinesR
1991             \else
1992                 \@writelinesinparR
1993                 \writtenlinesRtrue
1994             \fi
1995             \ifnum\l@dnumpstartsR>\l@dpscR
1996                 \writtenlinesRfalse
1997                 \ifbypstart@R
1998                     \ifnum\value{pstartR}<\value{pstartRold}
1999                     \else
2000                         \global\line@numR=0
2001                         \resetprevline@
2002                     \fi
2003                 \fi
2004                 \addtocounter{pstartR}{1}
2005                 \global\pstartnumRtrue
2006                 \l@dcalc@maxoftwo{\the\usernamecount{1@dmaxlinesinpar\the\l@dpscR}}%
2007                                 {\the\@donetotallinesR}%
2008                                 {\usernamecount{1@dmaxlinesinpar\the\l@dpscR}}%
2009                 \global\@donetotallinesR \z@
2010                 \global\advance\l@dpscR \@ne
2011             \fi
2012         \fi
2013     \fi}
2014

```

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>	1468, 1469, 1473, 1490, 1504
<code>\@M</code>	1479
<code>\@adv</code>	<u>339</u> , 608, 609
<code>\@afterindentfalse</code>	717
<code>\@arabic</code>	196, 197, 766, 769
<code>\@astanza@line</code>	1489, 1495, <u>1498</u>
<code>\@auxout</code>	1289, 1301, 1578
<code>\@chapter</code>	718
<code>\@cs@linesinparL</code>	1754, <u>1865</u>
<code>\@cs@linesinparR</code>	1754, <u>1865</u>
<code>\@cs@linesonpageL</code>	1762, 1822, <u>1878</u>
<code>\@cs@linesonpageR</code>	1762, 1822, <u>1878</u>
<code>\@currentlabel</code>	801, 834
<code>\@donereallinesL</code>	
.	<u>878</u> , 907, 1716, 1718, 1768
<code>\@donereallinesR</code>	
.	<u>878</u> , 942, 1721, 1723, 1770
<code>\@donetotallinesL</code>	<u>878</u> ,
908, 911, 1769, 1959, 1978, 1980	
<code>\@donetotallinesR</code>	<u>878</u> ,
943, 946, 1771, 1988, 2007, 2009	
<code>\@insertR</code>	1243–1245, 1258–1260
<code>\@l</code>	<u>262</u> , 577
<code>\@l@dttempcnta</code>	
.	412, 414, 416, 417, 421, 423,
425, 426, 996, 1035, 1036, 1038,	
1040, 1043, 1044, 1061–1065,	
1067, 1074, 1079, 1083, 1091,	
1096, 1100, 1133, 1136, 1138, 1142	
<code>\@l@dttempcntb</code>	155, 157, 159, 1026,
1027, 1074, 1079, 1083, 1091,	
1096, 1100, 1125, 1129, 1142,	
1150–1152, 1154, 1174–1176,	
1178, 1195–1197, 1199, 1330,	
1332, 1334, 1398–1400, 1402,	
1532–1536, 1540–1543, 1547–1550	
<code>\@l@reg</code>	311
<code>\@l@regR</code>	<u>262</u>
<code>\@lab</code>	531, 1280, 1292, <u>1316</u>
<code>\@lock</code>	894, 977
<code>\@lockR</code>	60, 284, 286, 288, 301,
446, 462, 463, 465, 466, 494,	
495, 497, 929, 960, 1002, 1004,	
1005, 1007, 1088, 1105, 1107, 1109	
<code>\@lopL</code>	<u>555</u> , 1892
<code>\@lopR</code>	<u>555</u> , 1895
<code>\@nameuse</code>	1426, 1430

- `\@nobreakfalse` 775, 808
`\@nobreaktrue` 773, 777, 806, 810
`\@oldnobreak` 773, 775, 806, 808, 848, 864
`\@pend` 546, 1716
`\@pendR` 546, 1721
`\@pstartfalse` 1689
`\@pstarttrue` 1689
`\@ref` 518, 581, 585
`\@ref@reg` 544
`\@schapter` 718
`\@set` 371, 615, 616, 1663, 1667
`\@tag` 645, 661
`\@temp` 1569
`\@templ@ed` 1389, 1390
`\@writelinesinparL` .. 1659, 1714, 1963
`\@writelinesinparR` .. 1660, 1714, 1992
`\@writelinesonpageL` . 1792, 1794, 1891
`\@writelinesonpageR` . 1812, 1814, 1891
`\@xloop` 1256
- A**
- `\absline@num` 405, 419, 438, 968
`\absline@numR` ... 58, 213, 264, 267,
 270, 402, 410, 431, 450, 484,
 512, 523, 951, 988, 989, 1026, 1242
`\actionlines@list`
 254, 257, 405, 419, 438
`\actionlines@listR`
 217, 232, 246, 249, 402,
 410, 431, 450, 484, 512, 1048, 1051
`\actions@list` . 258, 406, 426, 440, 442
`\actions@listR`
 . 217, 233, 250, 403, 417, 433,
 435, 452, 461, 486, 493, 513, 1052
`\add@inserts` 900
`\add@inserts@nextR` 1231
`\add@insertsR` 935, 1231
`\add@penaltiesL` 906, 1252
`\add@penaltiesR` 941, 1252
`\addtocontents` 1579–1581
`\addtocounter`
 . 850, 866, 1670, 1671, 1975, 2004
`\advancelabel@refs` 1287, 1299
`\advanceline` 607, 638
`\affixline@num` 898
`\affixline@numR` 933, 1058
`\affixpstart@numL` 897, 1164
`\affixpstart@numR` 932, 1164
`\affixside@note` 901
`\affixside@noteR` 936, 1376
- `\appto` 1382, 1383
`\apptocmd` 1609
`\araw@textfalse` 1701
`\araw@texttrue` 1701
`astanza` (environment) 9, 1471
`\AtBeginDocument` ... 1312, 1554, 1589
- B**
- `\ballast@count` 986, 991
`\bbl@main@language` 1605, 1606
`\bbl@set@language` 1596, 1597
`\beginnumbering` .. 7, 36, 724, 742, 780
`\beginnumberingR` ... 49, 104, 742, 813
`\bfseries` 766, 769
`\bypage@Rfalse` 124, 139, 144
`\bypage@Rtrue` 124, 134
`\bypstart@Rfalse` 124, 135, 145
`\bypstart@Rtrue` 124, 140
- C**
- `\c@ballast` 991
`\c@firstlinenumR` 164, 1131
`\c@firstsublinenumR` 168, 1126
`\c@linenumincrementR` 164, 1131
`\c@page` ... 577, 1843, 1849, 1852, 1859
`\c@pstartL` 766
`\c@pstartR` 769
`\c@sublinenumincrementR` .. 168, 1126
`\ch@ck@l@ckR` 1058
`\ch@cksub@l@ckR` 1058
`\ch@cksub@lockR` 1127
`\chapter` 704, 705, 713
`\chapterinpages` 697, 705, 715
`\chardef` 1468
`\check@goal` 1917, 1931, 1943
`\check@pstarts`
 1634, 1661, 1689, 1748, 1756, 1764
`\checkpageL` 1775, 1789, 1911
`\checkpageR` 1798, 1809, 1911
`\checkraw@text`
 1640, 1657, 1701, 1772, 1818
`\cleardoublepage` 1855
`\clearl@dleftpage` 1797, 1841
`\clearl@drightpage` 1817, 1841
`\cleartoevenpage` 1841
`\cleartol@devenpage` 1737, 1841
`\closeout` 568, 572
`\columnrulewidth` 5, 1684
`\Columns` 5, 1620
`\columnseparator` 5, 1654, 1684

\correcthangingL 903, 1449
 \correcthangingR 938, 1449
 \countLline 873, 884
 \countRline 873, 919
 \critext 643

D

\DeclareOption 8, 9
 \def@tempb 142
 \dimen 594, 595, 599–601, 605
 \divide 1063
 \do@actions 969
 \do@actions@fixedcodeR 995
 \do@actions@nextR 995
 \do@actionsR 952, 995
 \do@ballast 970
 \do@ballastR 953, 986
 \do@lineL 883, 1644, 1648, 1780
 \do@lineLhook 888, 915
 \do@lineR 918, 1645, 1650, 1801
 \do@lineRhook 915, 923
 \do@lockoff 481
 \do@lockoffL 505
 \do@lockoffR 481
 \do@lockon 446
 \do@lockonL 478
 \do@lockonR 446
 \dolistloop 1387
 \dummy@ref 527

E

\edfont@info 680, 683, 689, 692
 \edlabel 1278
 \edtext 659
 \eledmac@error 21, 24, 27, 29
 \eledmac@warning 1388
 \empty .. 78, 81, 246, 254, 653, 669,
 678, 687, 789, 822, 1048, 1130,
 1138, 1233–1235, 1246, 1257,
 1281, 1293, 1866, 1872, 1879, 1885
 \end@lemmas 653, 654, 669, 670
 \endashchar 1268
 \endgraf 844, 860, 1628, 1740
 \endline@num 534, 540
 \endlock 627, 1477, 1486, 1491
 \endnumbering 7, 39, 71, 108, 743
 \endnumberingR 52, 71, 93, 103, 116, 743
 \endpage@num 533, 540
 \endstanzaextra 1493
 \endsub 594

\endsubline@num 535, 541

environments:

 astanza 9, 1471
 Leftside 6, 722
 pages 5, 697
 pairs 5, 697
 Rightside 6, 740

\extensionchars .. 47, 66, 99, 113, 121

F

\f@x@l@cksR 1058
 \first@linenum@out@Rfalse .. 563, 569
 \first@linenum@out@Rtrue 563
 \firstlinenum 6, 173
 \firstsublinenum 6, 173
 \fix@page 307, 314
 \flag@end 578, 658, 674
 \flag@start 578, 650, 666
 \flush@notes 1673, 1827
 \flush@notesR 1255, 1674, 1828
 \fullstop .. 209, 1265, 1267, 1269, 1271

G

\get@linelistfile 242
 \get@nextboxL 1788, 1956
 \get@nextboxR 1808, 1956
 \getline@numL 893, 967
 \getline@numR 928, 950
 \getlinesfrompagelistL
 1760, 1820, 1878
 \getlinesfrompagelistR
 1761, 1821, 1878
 \getlinesfromparlistL ... 1752, 1865
 \getlinesfromparlistR ... 1753, 1865
 \gl@p 249, 250,
 257, 258, 654, 670, 682, 691,
 1051, 1052, 1239, 1243, 1258,
 1284, 1296, 1869, 1875, 1882, 1888
 \goalfraction 6, 1943

H

\hangingsymbol 10, 1440, 1446
 \hb@xt@ ... 896, 903, 910, 931, 938,
 945, 1652, 1783, 1785, 1804, 1806
 \hsize 799,
 832, 1652, 1783, 1785, 1804, 1806

I

\if@files w 1577
 \if@firstcolumn 1144, 1168, 1189, 1392

- \if@nobreak 772, 805
 - \if@pstarts 1635, 1689, 1749, 1765
 - \ifaraw@text ... 1642, 1701, 1774, 1819
 - \ifautopar 798, 831
 - \ifbypage@ 330
 - \ifbypage@R 124, 320, 1030
 - \ifbypstart@ 548, 1662, 1968
 - \ifbypstart@R ... 124, 552, 1666, 1997
 - \ifdim 595, 599, 601,
605, 1783, 1804, 1845, 1918, 1932
 - \iffirst@linenum@out@R 563, 567
 - \ifinserthangingsymbol .. 1438, 1452
 - \ifinserthangingsymbolR
..... 1436, 1444, 1462
 - \ifinstanzaL 720, 720, 1439, 1451
 - \ifinstanzaR 720, 721, 1445, 1461
 - \ifl@d@dash 1268
 - \ifl@d@elin 1270, 1271
 - \ifl@d@esl 1271
 - \ifl@d@pnum 1265, 1269
 - \ifl@d@ssub 1267
 - \ifl@d@pagefull 1791, 1811, 1911
 - \ifl@d@paging 11, 1450, 1460
 - \ifl@d@pairing 11, 75
 - \ifl@d@samelang 1565, 1643
 - \ifl@d@samepage 1778, 1800, 1911
 - \ifl@d@skipnumber 1121
 - \ifl@d@usedbabel 1563
 - \iflabelpstart 801, 834
 - \ifledplinenum 1266
 - \ifledRcol 11, 156, 178,
182, 186, 190, 229, 244, 308,
317, 341, 355, 372, 389, 401,
409, 430, 475, 502, 511, 520,
579, 589, 596, 602, 608, 615,
623, 628, 632, 637, 647, 663,
677, 1279, 1317, 1331, 1342,
1353, 1365, 1412, 1425, 1600, 1610
 - \ifnoteschanged@ 85
 - \ifnumberedpar@ ... 782, 815, 840,
856, 1341, 1352, 1364, 1411, 1424
 - \ifnumbering 37, 129, 778, 837
 - \ifnumberingR 50, 72, 95, 811, 853
 - \ifnumberline 954, 971, 1120
 - \ifnumberpstart ... 798, 831, 849, 865
 - \ifnumequal 1380
 - \ifnumgreater 1388
 - \ifodd 1154, 1178,
1199, 1402, 1843, 1849, 1852, 1859
 - \ifpst@rtedL 32, 786
 - \ifpst@rtedR 32, 819
 - \ifpstartnum 1209, 1214
 - \ifpstartnumR 1164
 - \ifshiftedpstarts 5, 1782, 1803, 1944
 - \ifsidepstartnum 798, 831, 1166, 1187
 - \ifsublines@ 207,
296, 340, 373, 380, 411, 420,
432, 439, 451, 485, 539, 541,
955, 972, 1037, 1124, 1319, 1323
 - \ifvbox 885, 920, 1705, 1708, 1957, 1986
 - \ifvmode 1286, 1298
 - \ifwrittenlinesL 1953, 1961
 - \ifwrittenlinesR 1954, 1990
 - \initnumbering@reg 45
 - \insert@count 517, 585, 648,
664, 1348, 1360, 1372, 1419, 1432
 - \insert@countR 518, 581,
647, 663, 1356, 1368, 1415, 1428
 - \inserthangingsymbolfalse 894
 - \inserthangingsymbolL 903, 1436
 - \inserthangingsymbolR 938, 1436
 - \inserthangingsymbolRfalse 929
 - \inserthangingsymbolRtrue 929
 - \inserthangingsymboltrue 894
 - \insertlines@listR
.... 78, 217, 231, 523, 1235, 1239
 - \inserts@list
788, 1347, 1359, 1371, 1418, 1431
 - \inserts@listR
.. 821, 1230, 1233, 1243, 1257,
1258, 1344, 1355, 1367, 1414, 1427
 - \instanzaLfalse 1681, 1834
 - \instanzaLtrue 731
 - \instanzaRfalse 1682, 1835
 - \instanzaRtrue 753
 - \interlinepenalty 1479
 - \itemcount@ 1378, 1380, 1385, 1388
- L**
- \l@d@nums 680, 683, 689, 692
 - \l@d@set 388, 623, 624
 - \l@dbbl@set@language 1574, 1597
 - \l@dbfnote 1410
 - \l@dc@maxchunks 794, 796,
827, 829, 1522, 1532, 1540, 1547
 - \l@d@calc@maxoftwo
..... 1754, 1898, 1977, 2006
 - \l@d@calc@minoftwo .. 1762, 1822, 1898
 - \l@d@calcnun 1058
 - \l@d@checklang 1567, 1641

- \l@dchset@num 263, 266, 388
- \l@dcsnote 1340
- \l@dcsnotetext 1387, 1390
- \l@demptyd@ta 889, 924
- \l@dend@stuff ... 48, 67, 100, 114, 122
- \l@dgetline@margin 154
- \l@dgetsidenote@margin 1329
- \l@dld@ta 899, 934,
1145, 1157, 1169, 1181, 1190, 1202
- \l@dleftbox
.. 870, 895, 910, 1653, 1783, 1785
- \l@dlinenumR 199
- \l@dlsn@te 902, 937
- \l@dlsnote 1340
- \l@dmake@labels 1302
- \l@dmake@labelsR 1290, 1306
- \l@dminpagelines
.... 1725, 1763, 1823, 1919, 1933
- \l@dnumstartL . 41, 793, 794, 796,
798, 1526, 1558, 1623, 1624,
1678, 1692, 1734, 1735, 1832, 1966
- \l@dnumstartR . 54, 826, 827, 829,
831, 1526, 1559, 1623, 1624,
1679, 1695, 1734, 1735, 1833, 1995
- \l@doldbbl@set@language 1596
- \l@doldselectlanguage 1595, 1599, 1604
- \l@dpagefullfalse 1911
- \l@dpagefulltrue 1911
- \l@dpagefalse 13, 699, 712
- \l@dpageingtrue 707
- \l@dpairingfalse 11, 701, 711
- \l@dpairingtrue 698, 706
- \l@dpscL 885, 890, 1528, 1560, 1632,
1638, 1676, 1692, 1705, 1744,
1750, 1755, 1758, 1766, 1830,
1957, 1959, 1966, 1977, 1979, 1981
- \l@dpscR 920, 925, 1529, 1561,
1633, 1639, 1677, 1695, 1708,
1745, 1751, 1759, 1767, 1831,
1986, 1988, 1995, 2006, 2008, 2010
- \l@dird@ta 903, 938,
1147, 1155, 1171, 1179, 1192, 1200
- \l@drightbox
.. 870, 930, 945, 1655, 1804, 1806
- \l@drrsn@te 904, 939
- \l@drrsnote 1340
- \l@dsamelangfalse 1565, 1568
- \l@dsamelangtrue 1565, 1571
- \l@dsamepagefalse 1911
- \l@dsamepagetrue 1911
- \l@dsetupmaxlinecounts .. 1539, 1556
- \l@dsetuprawboxes 1531, 1555
- \l@dskipnumberfalse 1122
- \l@dskipnumbertrue 1018
- \l@dunhbox@line 903, 938
- \l@dusedbabelfalse 1563, 1592
- \l@dusedbabeltrue 1563, 1594
- \l@duselanguage
.... 1584, 1647, 1649, 1776, 1799
- \l@dzeromaxlinecounts ... 1539, 1557
- \l@dzeropenalties 843, 859, 1627, 1739
- \l@pscL 1528
- \l@pscR 1528
- \label@refs
1282, 1284, 1290, 1294, 1296, 1302
- \labelref@list 1293, 1296, 1324
- \labelref@listR 1276, 1281, 1284, 1320
- \languagename .. 1575, 1576, 1578–1581
- \last@page@num 328, 334
- \last@page@numR 314
- \lastbox 892, 927
- \lastskip 594, 600
- \lcolwidth 5, 6, 15, 708, 799, 896, 910
- \led@err@BadLeftRightPstarts ...
..... 23, 1624, 1735
- \led@err@LeftOnRightPage ... 26, 1853
- \led@err@LineationInNumbered ... 130
- \led@err@NumberingNotStarted ... 89
- \led@err@NumberingShouldHaveStarted
..... 102
- \led@err@NumberingStarted 38, 51
- \led@err@PendNoPstart 841, 857
- \led@err@PendNotNumbered ... 838, 854
- \led@err@PstartInPstart ... 783, 816
- \led@err@PstartNotNumbered . 779, 812
- \led@err@RightOnLeftPage ... 26, 1860
- \led@err@TooManyPstarts 20, 795, 828
- \led@mess@NotesChanged 86
- \led@mess@SectionContinued
..... 98, 112, 120
- \led@warn@BadAction 1020
- \led@warn@BadAdvancelineLine 358, 364
- \led@warn@BadAdvancelineSubline .
..... 344, 350
- \led@warn@BadLineation 147
- \led@warn@BadSetline 613
- \led@warn@BadSetlinenum 621
- \led@warn@DuplicateLabel 1308
- \ledllfill 903, 938
- \ledRcolfalse 14, 723, 755

- \ledRcoltrue 741
 - \ledrlfill 903, 938
 - \ledsavedprintlines 8, 1263
 - \ledstrutL 1783, 1785, 1838
 - \ledstrutR 1804, 1806, 1838
 - \ledthegoal 1918, 1932, 1943
 - \leftlinenumR 199, 1145, 1157
 - \leftpstartnumL 1164
 - \leftpstartnumR 1164
 - Leftside (environment) 6, 722
 - \Leftsidehook 729, 735
 - \Leftsidehookend 734, 735
 - \line@list 687, 691
 - \line@list@stuff 47, 113
 - \line@list@stuffR ... 66, 99, 121, 565
 - \line@listR . 81, 217, 230, 541, 678, 682
 - \line@margin 159, 1174
 - \line@marginR 152, 1150, 1195
 - \line@num . 331, 362, 363, 365, 383,
394, 395, 423, 548, 978, 1322, 1971
 - \line@numR 59, 206,
213, 268, 302, 321, 356, 357,
359, 376, 390, 391, 414, 534,
538, 552, 961, 1031, 1040, 1129,
1131, 1133, 1134, 1318, 1388, 2000
 - \lineation 750
 - \lineationR 128, 750
 - \linenum@out 584,
592, 597, 603, 609, 616, 624,
629, 633, 1292, 1663, 1716, 1892
 - \linenum@outR 562, 568,
570, 572, 573, 577, 580, 590,
596, 602, 608, 615, 623, 628,
632, 637, 1280, 1667, 1721, 1895
 - \linenumberlist 1130, 1134
 - \linenumincrement 6, 173
 - \linenummargin 152
 - \linenumr@p 1266, 1270, 1318, 1322
 - \linenumrepR 196, 206
 - \linenumsep
. 201, 203, 1211, 1214, 1223, 1226
 - \linesinpar@listL
..... 222, 238, 549, 1866, 1869
 - \linesinpar@listR
..... 222, 234, 553, 1872, 1875
 - \linesonpage@listL 239, 557, 1879, 1882
 - \linesonpage@listR 235, 560, 1885, 1888
 - \list@clear
. 230–235, 238, 239, 241, 788, 821
 - \list@clearing@reg 237
 - \list@create
... 217–220, 222–224, 1230, 1276
 - \lock@disp 1090, 1094, 1099
 - \lock@off 472, 473, 481, 632, 633
 - \lock@on 628, 629
- M**
- \maxchunks 4, 1522
 - \maxlinesinpar@list 222, 241
 - \memorydump 8, 728, 746
 - \memorydumpL 107, 728
 - \memorydumpR 107, 746
 - \message 46, 65
 - \multiply 1064
- N**
- \n@num 509, 637
 - \n@num@reg 515
 - \namebox 885, 890, 920,
925, 1506, 1705, 1708, 1957, 1986
 - \NeedsTeXFormat 2
 - \new@line 903
 - \new@lineR 576, 938
 - \newbox 761, 870, 871, 1507
 - \newcounter 164,
166, 168, 170, 764, 765, 767, 768
 - \newif . 5, 12, 33, 124, 125, 563, 720,
721, 1218, 1436, 1563, 1565,
1689, 1701, 1911, 1913, 1953, 1954
 - \newnamebox 1506, 1534, 1535
 - \newnamecount 1517, 1542
 - \newwrite 562
 - \next@action 258
 - \next@actionline 255, 257
 - \next@actionlineR
.. 247, 249, 989, 1027, 1049, 1051
 - \next@actionR 250, 990,
1028, 1029, 1034, 1035, 1043, 1052
 - \next@insert 789
 - \next@insertR
822, 1234, 1237, 1239, 1242, 1246
 - \next@page@num 335, 406
 - \next@page@numR 63, 271, 273, 325, 403
 - \no@expands 645, 661
 - \normal@pars 74, 792, 825
 - \normalbfnoteX 1423
 - \noteschanged@true
..... 79, 82, 679, 688, 1236
 - \num@lines 844, 1628, 1740
 - \num@linesR 760, 860, 1629, 1741

- `\numberedpar@true` 800, 833
`\numberingRfalse` 73
`\numberingRtrue` 56, 93, 117
`\numberingtrue` 43, 109
`\numberpstartfalse` 9
`\numberpstarttrue` 9
`\numdef` 1378, 1385
`\numlabfont` 206
`\numpagelinesL`
 1725, 1781, 1792, 1796, 1919
`\numpagelinesR`
 1725, 1802, 1812, 1816, 1933
- O**
- `\oldchapter` 704, 713
`\oldstanza` 730, 731, 733, 752, 753, 756
`\one@line` 890, 892, 903
`\one@lineR` 760, 925, 927, 938
`\openout` 570, 573
- P**
- `\p@pstartL` 802
`\p@pstartR` 835
`\page@action` 272, 400, 528
`\page@num` 253, 333, 1176, 1400
`\page@numR` 226, 245, 323,
 533, 538, 1029, 1152, 1197, 1388
`\pagegoal` 1951
`\Pages` 5, 1729
`pages` (environment) 5, 697
`\pagetotal` 1845, 1918, 1932
`pairs` (environment) 5, 697
`\par@line` 845, 1630, 1742
`\par@lineR` 760, 861, 1631, 1743
`\pausenumbering` 744
`\pausenumberingR` 92, 744
`\pend` 6, 727, 749, 784, 1492
`\pendL` 727, 837
`\pendR` 749, 817, 853
`\prevgraf`
 . 844, 860, 1628, 1629, 1740, 1741
`\printlines` 1274
`\printlinesR` 8, 1263
`\ProcessOptions` 10
`\protected@edef` 801, 834
`\protected@write` ... 1289, 1301, 1578
`\ProvidesPackage` 3
`\pst@rtedLfalse` 32, 42
`\pst@rtedLtrue` 110, 790
`\pst@rtedRfalse` 34, 55, 76
`\pst@rtedRtrue` 96, 118, 823
`\pstart` 6, 21, 25, 725, 748, 1494
`\pstartL` 725, 763
`\pstartnumfalse` 1211, 1216
`\pstartnumRfalse` 1223, 1228
`\pstartnumRtrue` 1219, 1637, 2005
`\pstartnumtrue` 1636, 1976
`\pstartR` 748, 763
- R**
- `\Rcolwidth` 5, 6, 15, 709, 832, 931, 945
`\read@linelist` 228, 566
`\rem@inder` 1134, 1136–1138
`\resetprevline@` 1664, 1668, 1972, 2001
`\resumenumbering` 745
`\resumenumberingR` 92, 745
`\rightlinenumR` 199, 1147, 1155
`\rightpstartnumL` 1164
`\rightpstartnumR` 1164
`Rightside` (environment) 6, 740
`\Rightsidehook` 735, 751
`\Rightsidehookend` 735, 757
`\rlap` 1147, 1155, 1171, 1179, 1192, 1200
`\Rlineflag` 8, 194, 206, 1266, 1270, 1310
`\rule` 1685
- S**
- `\sc@n@list` 1135, 1137
`\secdef` 718
`\section@num` 44, 46, 47, 111–113
`\section@numR`
 ... 30, 57, 65, 66, 97–99, 119–121
`\select@language` ... 1576, 1578–1581
`\selectlanguage` 1584
`\set@line` 646, 662, 676
`\set@line@action`
 265, 369, 378, 385, 408, 530
`\setl@dlp@rbox` 1393, 1405
`\setl@drp@rbox` 1395, 1403
`\setline` 611
`\setlinenum` 619
`\setnamebox` 798, 831, 1506
`\setprintlines` 1264
`\shiftedpstartfalse` 7
`\shiftedpstarttrue` 6, 8, 9
`\shiftedversesfalse` 7
`\shiftedversestrue` 6
`\showlemma` 652, 668
`\sidenote@margin` 1334, 1338
`\sidenote@marginR` 1327, 1398

- `\sidenotecontent@`
 1377, 1382, 1383, 1393, 1395, 1405
`\sidenotecontent@t` 1403
`\sidenotemargin` 1327
`\sidenotessep` 1383
`\skip@lockoff` 473, 481
`\skipnumbering` 9, 636
`\skipnumbering@reg` 640
`\smash` 1685
`\splittopskip` 887, 922
`\stanza` ... 730, 731, 733, 752, 753, 756
`\stanza@count` 1474, 1488, 1499
`\stanza@hang` 1476, 1501
`\stanzaindentbase` .. 1454, 1464, 1499
`\startlock` 627
`\startstanzahook` 1472
`\startsub` 594
`\sub@action` 281, 429, 529
`\sub@change` 64, 275, 276, 282
`\sub@lock` 973
`\sub@lockR` 61, 290, 292, 294,
 297, 447, 453, 454, 456, 457,
 487, 488, 490, 956, 1010, 1012,
 1013, 1015, 1071, 1111, 1113, 1115
`\sub@off` 602, 603
`\sub@on` 596, 597
`\subline@num` 208, 331, 348,
 349, 351, 381, 421, 974, 979, 1323
`\subline@numR` 209,
 213, 298, 302, 321, 342, 343,
 345, 374, 412, 535, 539, 957,
 962, 1031, 1038, 1125, 1126, 1319
`\sublinenumincrement` 6, 173
`\sublinenumr@p` . 1267, 1271, 1319, 1323
`\sublinenumrepR` 196, 209
`\sublines@false` 62, 279, 1000
`\sublines@true` 277, 998
`\sublock@disp` 1073, 1077, 1082
`\symplinenum` 1266
`\sza@penalty` 1483, 1487
- T**
- `\textwidth` 16, 18, 708, 709
`\theledlanguageL` 1569, 1584, 1647, 1776
`\theledlanguageR` 1569, 1584, 1649, 1799
`\thepage` 577, 1290, 1302
`\thepstart` 726, 747
`\thepstartL`
 . 9, 726, 766, 798, 802, 1210, 1215
- `\thepstartR`
 . 9, 747, 769, 831, 835, 1222, 1227
`\thr@@` .. 456, 465, 488, 495, 1005, 1013
`\topskip` 1845
- U**
- `\unhbox` 1511,
 1653, 1655, 1783, 1785, 1804, 1806
`\unhnamebox` 1506
`\unvbox` 892, 927, 1513
`\unvnamebox` 1506
`\usenamecount`
 . 1475, 1482, 1517, 1549, 1755,
 1959, 1977, 1979, 1988, 2006, 2008
- V**
- `\value` 787, 820,
 1621, 1622, 1730, 1731, 1969, 1998
`\vbadness` 886, 921
`\vbfnoteX` 1426, 1430
`\vbox` 798, 831
`\vl@dbfnote` 1413, 1417
`\vl@dcsnote` 1366, 1370
`\vl@dlsnote` 1343, 1346
`\vl@drsnote` 1354, 1358
`\vsplit` 890, 925
- W**
- `\wd` 903, 938
`\writtenlinesLfalse` 1746, 1967
`\writtenlinesLtrue` 1964
`\writtenlinesRfalse` 1747, 1996
`\writtenlinesRtrue` 1993
- X**
- `\x@lemma` 654–656, 670–672
`\xpg@main@language` 1614, 1615
`\xpg@set@language` 1609, 1613
`\xright@appenditem`
 402, 403, 405, 406, 410,
 417, 419, 426, 431, 433, 435,
 438, 440, 442, 450, 452, 461,
 484, 486, 493, 512, 513, 523,
 537, 549, 553, 557, 560, 1318,
 1322, 1343, 1346, 1354, 1358,
 1366, 1370, 1413, 1417, 1426, 1430
- Z**
- `\zz@@@` 1282, 1294

Change History

v0.1		\Columns	59
General: First public release	1	\Pages: Added \l@duselanguage to \Pages	63
v0.10		v0.3	
General: \edlabel commands on the right side are now correctly indicated.	1	General: Reorganize for ledarab . .	1
\edlabel commands which start a paragraph are now put in the right place.	1	\affixline@numR: Changed \affixline@numR to match new eledmac	42
v0.11		\do@actions@nextR: Used \do@actions@fixedcode in \do@actionsR	41
General: Change \do@lineL and \do@lineR to allow line num- bering by pstart(like in eledmac 0.15).	38	\do@lineL: Added \do@linehook to \do@lineL	38
Lineation can be by pstart (like in eledmac 0.15).	16	Simplified \do@lineL by using macros for some common code	38
New management of hang- ingsymbol insertion, preventing undesirable insertions.	52	\do@lineR: Changed \do@lineR similarly to \do@lineL	39
Prevent shift of column separator when a verse is hanged	53	\do@lineRhook: Added \do@linehook and \do@lineRhook	39
\affixline@numR: Changed \affixline@numR to allow to disable line numbering (like in eledmac 0.15).	42	Leftside: Added hooks into Left- side environment	33
\Columns: Line numbering by pstart.	60	\flag@end: Removed extraneous spaces from \flag@end	29
\get@nextboxR: Change \get@nextboxL and \get@nextboxR to allow to disable line numbering (like in eledmac 0.15).	68	\ifledRcol: Moved \ifl@dpairing to eledmac	13
Pstart number can be printed in side	68	\ifpst@rtedR: Moved \ifpst@rtedL to eledmac	14
v0.12		\l@ddlinenumR: Simplified \leftlinenumR and \rightlinenumR by introducing \l@ddlinenumR	18
General: New new management of hangingsymbol insertion, pre- venting undesirable insertions.	52	\l@dnumpstartsR: Moved \l@dnumpstartsL to eledmac .	55
v0.2		\ledsavedprintlines: Simpli- fied \printlinesR by using \setprintlines	48
General: Added section of babel re- lated code	56	\ledstrutR: Added \ledtrutL and \ledstrutR	65
Fix babel problems	1	\normalbfnoteX: Removed extraneous spaces from \normalbfnoteX	52
\Columns: Added \l@dchecklang and \l@duselanguage to		\Pages: Added \ledstrutL to \Pages	63
		Added \ledstrutR to \Pages .	64

	<code>\Rightsidehookend:</code>	Added	bol on each hanging of verses,
	<code>\Leftsidehook, \Leftsidehookend,</code>		like in the french typogra-
	<code>\Rightsidehook and \Rightsidehookend</code>		phy. Redefine the commande
	34	<code>\hangingsymbol</code> to define the
	<code>\sublinenumrepR:</code>	Added	character. 1
	<code>\linenumrepR and \sublinenumrepR</code>	v0.9	
	18	General: Possibility to number
v0.3a	General: Minor <code>\linenummargin</code>		<code>\pstart</code> 9
	fix	1	Possibility to number the
	<code>\line@marginR:</code> Don't just		<code>pstart</code> with the commands
	set <code>\line@marginR</code> in		<code>\numberpstarttrue</code> 1
	<code>\linenummargin</code>	17	<code>\ifledRcol:</code> Moved <code>\iflledRcol</code>
v0.3b			and <code>\ifnumberingR</code> to <code>eledmac</code> 13
	General: Improved parallel page		
	balancing	1	v0.9.1
	<code>\Pages:</code> Added <code>\l@dminpagelines</code>		General: The numbering of the
	calculation for succeeding page		<code>pstarts</code> restarts on each
	pairs	64	<code>\beginnumbering</code> 1
v0.3c			v0.9.2
	General: Compatibility with Poly-		General: Debug : with <code>\Columns</code> ,
	glossia	1	the hanging indentation now
v0.4			runs on the left columns and the
	General: No more <code>ledparpatch</code> . All		hanging symbol is shown only
	patches are now in the main		when <code>\stanza</code> is used. 1
	file.	1	v0.9.3
v0.5			General: <code>\thepstartL</code> and
	General: Corrections about		<code>\thepstartR</code> use now
	<code>\section</code> and other titles in		<code>\bfseries</code> and not <code>\bf</code> , which
	numbered sections	1	is deprecated and makes con-
v0.6			licts with memoir class. 1
	General: Be able to use <code>\chapter</code> in		v1.0
	parallel pages.	1	General: Compatibility with <code>eled-</code>
v0.7			<code>mac</code> . Change name to <code>eledpar</code> . . 1
	General: Option 'shiftedverses'		Debug in lineation by <code>pstart</code> ... 16
	which make there is no blank		v1.0.1
	between two parallel verses with		General: Correction on <code>\numberonlyfirstinline</code>
	inequal length.	1	with lineation by <code>pstart</code> or by
v0.8			page. 1
	General: Possibility to have a sym-		v1.1
			General: <code>Shiftedverses</code> becomes
			<code>shiftedpstarts</code> 1