

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

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

Abstract

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

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

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

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

Contents

1 Introduction	3
2 The eledpar package	4
2.1 General	4
3 Parallel columns	5
4 Facing pages	6
5 Left and right texts	6
6 Numbering text lines and paragraphs	8

*This file (**eledpar.dtx**) has version number v1.7.0, last revised 2014/04/14.

[†]herries dot press at earthlink dot net

[‡]maieul at maieul dot net

7 Verse	9
8 Parallel ledgroups	11
8.1 Parallel ledgroups and <code>setspace</code> package	12
9 Implementation overview	14
10 Preliminaries	14
10.1 Messages	15
11 Sectioning commands	15
12 Line counting	18
12.1 Choosing the system of lineation	18
12.2 Line-number counters and lists	21
12.3 Reading the line-list file	22
12.4 Commands within the line-list file	23
12.5 Writing to the line-list file	31
13 Marking text for notes	34
14 Parallel environments	36
15 Paragraph decomposition and reassembly	37
15.1 Boxes, counters, <code>\pstart</code> and <code>\pend</code>	38
15.2 Processing one line	41
15.3 Line and page number computation	43
15.4 Line number printing	46
15.5 Pstart number printing in side	48
15.6 Add insertions to the vertical list	50
15.7 Penalties	50
15.8 Printing leftover notes	51
16 Footnotes	52
16.1 Normal footnote formatting	52
17 Cross referencing	52
18 Side notes	54
19 Familiar footnotes	56
20 Verse	57
21 Naming macros	59
22 Counts and boxes for parallel texts	59

<i>List of Figures</i>	3
23 Fixing babel	61
24 Parallel columns	63
25 Parallel pages	66
26 Page break/no page break, depending on the specific line	76
27 Parallel ledgroup	77
28 The End	80
Appendix A Some things to do when changing version	81
Appendix A.1 Migration to eledpar 1.4.3	81
References	81
Index	81
Change History	91

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 9 through 28); 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 9, 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 `pdflatex` or `xelatex`. 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.

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.

```
\firstlinenum
\linenumincrement
\firstsublinenum
\sublinenumincrement
\firstlinenum*
\linenumincrement*
\firstsublinenum*
\sublinenumincrement*
```

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

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. The starred versions change both left and right numbering schemes.

`\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 `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 followed 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` 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}}
```

`\numberpstarttrue` 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`

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 `\sza000`' it is because you have forgotten to use `\setstanzaindents` to set the stanza indents.

`\skipnumbering` The command `\skipnumbering` when inserted in a line of parallel text causes the numbering of that particular line to be skipped. This can be useful if you are putting some kind of marker (even if it is only a blank line) between stanzas.

Remember, parallel texts must be numbered and this provides a way to slip in an ‘unnumbered’ line.

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

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

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

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

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

  \end{astanza}
  ...
```

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

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

```

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

```

`\hangingsymbol` Like in `eledmac`, you could redefine the command `\hangingsymbol` to insert a character in each hanged line. If you use it, you must run \LaTeX two time. Example for the french typographie

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

You can also use it to force hanging verse to be flush right:

```
\renewcommand{\hangingsymbol}{\protect\hfill}
```

When you use `\lednopb` make sure to use it on both sides in the corresponding verses to keep the pages in sync.

8 Parallel ledgroups

You can also make parallel ledgroups (see the documentation of `eledmac` about ledgroups). To do it you have:

- To load `eledpar` package with the `parledgroup` option, or to add `\`.
- To push each ledgroup between `\pstart... \pend` command.

See the following example:

```

\begin{pages}
\begin{Leftside}
\beginnumbering
\pstart
\begin{ledgroup}
ledgroup content
\end{ledgroup}
\pend
\pstart
\begin{ledgroup}
ledgroup content
\end{ledgroup}
\pend
\endnumbering
\end{Leftside}

```

```

\begin{Rightside}
  \beginnumbering
  \pstart
    \begin{ledgroup}
      ledgroup content
    \end{ledgroup}
  \pend
  \pstart
    \begin{ledgroup}
      ledgroup content
    \end{ledgroup}
  \pend
  \endnumbering
\end{Rightside}
\Pages
\end{pages}

```

You can add sectioning a sectioning command, following this scheme:

```

\begin{..side}
  \beginnumbering
  \pstart
    \section{First ledgroup title}
  \pend
  \pstart
    \begin{ledgroup}\skipnumbering
      ledgroup content
    \end{ledgroup}
  \pend
  \pstart
    \section{Second ledgroup title}
  \pend
  \pstart
    \begin{ledgroup}\skipnumbering
      ledgroup content
    \end{ledgroup}
  \pend
  \endnumbering
\end{..side}

```

8.1 Parallel ledgroups and **setspace** package

. If you use the **setspace** package and want your notes in parallel ledgroups ledgroups to be single-spaced (not half-spaced or double-spaced), just add to your preamble:

```
\let\parledgroupnotespacing\singlespacing
```

In effect, to have correct spacing, don't change the font size of your notes.

9 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`.

10 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}[2014/04/14 v1.7.0 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 \DeclareOption{parledgroup}{\parledgrouptrue}
11 \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.
12 \l@dpairingfalse
13 \newif\ifl@dpaging
14 \l@dpagingfalse
15 \ledRcolfalse

\lcolwidth The widths of the left and right parallel columns (or pages).
\rcolwidth 16 \newdimen\lcolwidth
17 \lcolwidth=0.45\textwidth
18 \newdimen\rcolwidth
19 \rcolwidth=0.45\textwidth
20
```

10.1 Messages

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

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

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

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

11 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.

```

31 \newcount\section@numR
32 \section@numR=\z@

\ifpst@rtedL \ifpst@rtedL is set FALSE at the start of left side numbering, and similarly for
\ifpst@rtedR \ifpst@rtedR. \ifpst@rtedL is defined in eledmac.
33 \pst@rtedLfalse
34 \newif\ifpst@rtedR
35 \pst@rtedRfalse
36

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

\beginnumberingR This is the right text equivalent of \beginnumbering, and begins a section of
numbered text.
50 \newcommand*\beginnumberingR}{%
51   \ifnumberingR
52     \led@err@NumberingStarted
53   \endnumberingR
54 \fi
55 \global\l@dnumpstartsR \z@
56 \global\pst@rtedRfalse
57 \global\numberingRtrue
58 \global\advance\section@numR \@ne
59 \global\absline@numR \z@
60 \gdef\normal@page@breakR{}
61 \gdef\l@prev@pbR{}
62 \gdef\l@prev@nopbR{}
63 \global\line@numR \z@
64 \global\@lockR \z@
65 \global\sub@lockR \z@
66 \global\sublines@false
67 \global\let\next@page@numR\relax
68 \global\let\sub@change\relax
69 \message{Section \the\section@numR R }%
70 \line@list@stuffR{\jobname.\extensionchars\the\section@numR R}%

```



```

71 \l@dend@stuff
72 \setcounter{pstartR}{1}
73 \begingroup
74 \initnumbering@sectcmd
75 \initnumbering@sectcountR
76 }
77

```

`\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.

```

78 \def\endnumberingR{%
79   \ifnumberingR
80     \global\numberingRfalse
81     \normal@pars
82     \ifl@dpairing
83       \global\pst@rtedRfalse
84     \else
85       \ifx\insertlines@listR\empty\else
86         \global\noteschanged@true
87       \fi
88       \ifx\line@listR\empty\else
89         \global\noteschanged@true
90       \fi
91     \fi
92     \ifnoteschanged@
93       \led@mess@NotesChanged
94     \fi
95   \else
96     \led@err@NumberingNotStarted
97   \fi\endgroup}
98

```

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

```

99 \newcounter{chapterR}
100 \newcounter{sectionR}
101 \newcounter{subsectionR}
102 \newcounter{subsubsectionR}
103 \newcommand{\initnumbering@sectcountR}{
104   \let\c@chapter\c@chapterR
105   \let\c@section\c@sectionR
106   \let\c@subsection\c@subsectionR
107   \let\c@subsubsection\c@subsubsectionR
108 }

```

```

\pausenumberingR These are the right text equivalents of \pausenumbering and \resumenumbering.
\resumenumberingR 109 \newcommand*{\pausenumberingR}{%
110 \endnumberingR\global\numberingRtrue}
111 \newcommand*{\resumenumberingR}{%
112 \ifnumberingR
113 \global\pst@rtedRtrue
114 \global\advance\section@numR \@ne
115 \led@mess@SectionContinued{\the\section@numR R}%
116 \line@list@stuffR{\jobname.\extensionchars\the\section@numR R}%
117 \l@dend@stuff
118 \else
119 \led@err@numberingShouldHaveStarted
120 \endnumberingR
121 \beginnumberingR
122 \fi}
123

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

124 \newcommand*{\memorydumpL}{%
125 \endnumbering
126 \numberingtrue
127 \global\pst@rtedLtrue
128 \global\advance\section@num \@ne
129 \led@mess@SectionContinued{\the\section@num}%
130 \line@list@stuff{\jobname.\extensionchars\the\section@num}%
131 \l@dend@stuff}
132 \newcommand*{\memorydumpR}{%
133 \endnumberingR
134 \numberingRtrue
135 \global\pst@rtedRtrue
136 \global\advance\section@numR \@ne
137 \led@mess@SectionContinued{\the\section@numR R}%
138 \line@list@stuffR{\jobname.\extensionchars\the\section@numR R}%
139 \l@dend@stuff}
140

```

12 Line counting

12.1 Choosing the system of lineation

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.

```

\ifbypstart@R The \ifbypage@R and \ifbypstart@R flag specifie the current lineation system:
\byppstart@Rtrue
\byppstart@Rfalse
\ifbypage@R
\bypage@Rtrue
\bypage@Rfalse

```

- 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.

```

141 \newif\ifbypage@R
142 \newif\ifbypstart@R
143 \bypage@Rfalse
144 \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`.

```

145 \newcommand*{\lineationR}[1]{%
146   \ifnumbering
147     \led@err@LineationInNumbered
148   \else
149     \def\@tempa{#1}\def\@tempb{page}%
150     \ifx\@tempa\@tempb
151       \global\bypage@Rtrue
152       \global\bypstart@Rfalse
153     \else
154       \def\@tempb{pstart}%
155       \ifx\@tempa\@tempb
156         \global\bypage@Rfalse
157         \global\bypstart@Rtrue
158       \else
159         \def\@tempb{section}
160         \ifx\@tempa\@tempb
161           \global\bypage@Rfalse
162           \global\bypstart@Rfalse
163         \else
164           \led@warn@BadLineation
165         \fi
166       \fi
167     \fi
168   \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.

```

169 \newcount\line@marginR
170 \renewcommand*{\linenummargin}[1]{%
171   \l@getline@margin{#1}%
172   \ifnum\@l@tempcntb>\m@ne

```

```

173 \ifledRcol
174 \global\line@marginR=\@l@tempcntb
175 \else
176 \global\line@margin=\@l@tempcntb
177 \fi
178 \fi}}

```

By default put right text numbers at the right.

```

179 \line@marginR=\@ne
180

```

`\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.

```

181 \newcounter{firstlinenumR}
182 \setcounter{firstlinenumR}{5}
183 \newcounter{linenumincrementR}
184 \setcounter{linenumincrementR}{5}

```

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

```

185 \newcounter{firstsublinenumR}
186 \setcounter{firstsublinenumR}{5}
187 \newcounter{sublinenumincrementR}
188 \setcounter{sublinenumincrementR}{5}
189

```

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

`\linenumincrement`

`\firstsublinenum`

`\sublinenumincrement`

```

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

```

```

207 \ifledRcol \setcounter{sublinenumincrementR}{#1}%
208 \else \setcounter{sublinenumincrement}{#1}%
209 \fi}
210 \WithSuffix\newcommand\firstlinenum*[1]{\setcounter{firstlinenumR}{#1}\setcounter{firstlinenum}{#1}}
211 \WithSuffix\newcommand\linenumincrement*[1]{\setcounter{linenumincrementR}{#1}\setcounter{linenumincr
212 \WithSuffix\newcommand\firstsublinenum*[1]{\setcounter{subfirstlinenumR}{#1}\setcounter{subfirstlinen
213 \WithSuffix\newcommand\sublinenumincrement*[1]{\setcounter{sublinenumincrementR}{#1}\setcounter{subli

```

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

```

214 \newcommand*\Rlineflag{R}
215

```

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

```

216 \newcommand*\linenumrepR[1]{\@arabic{#1}}
217 \newcommand*\sublinenumrepR[1]{\@arabic{#1}}
218

```

`\leftlinenumR` `\leftlinenumR` and `\rightlinenumR` are the macros that are called to print the right text's marginal line numbers. Much of the code for these is common and is maintained in `\l@dlinenumR`.

```

219 \newcommand*\leftlinenumR{%
220   \l@dlinenumR
221   \kern\linenumsep}
222 \newcommand*\rightlinenumR{%
223   \kern\linenumsep
224   \l@dlinenumR}
225 \newcommand*\l@dlinenumR{%
226   \numlabfont\linenumrepR{\line@numR}\Rlineflag%
227   \ifsublines@
228     \ifnum\subline@num>\z@
229       \unskip\fullstop\sublinenumrepR{\subline@numR}%
230     \fi
231   \fi}
232

```

12.2 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.

```

233 \newcount\line@numR
234 \newcount\subline@numR

```

```

235 \newcount\absline@numR
236
\line@listR Now we can define the list macros that will be created from the line-list file. They
\insertlines@listR are directly analagous to the left text ones. The full list of action codes and their
\actionlines@listR meanings is given in the eledmac manual.
\actions@listR Here are the commands to create these lists:
237 \list@create{\line@listR}
238 \list@create{\insertlines@listR}
239 \list@create{\actionlines@listR}
240 \list@create{\actions@listR}
241
\linesinpar@listL In order to synchronise left and right chunks in parallel processing we need to know
\linesinpar@listR how many lines are in each left and right text chunk, and the maximum of these
\maxlinesinpar@list for each pair of chunks.
242 \list@create{\linesinpar@listL}
243 \list@create{\linesinpar@listR}
244 \list@create{\maxlinesinpar@list}
245
\page@numR The right text page number.
246 \newcount\page@numR
247

```

12.3 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.
248 \renewcommand*{\read@linelist}[1]{%
We do do different things depending whether or not we are processing right text
249 \ifledRcol
250 \list@clear{\line@listR}%
251 \list@clear{\insertlines@listR}%
252 \list@clear{\actionlines@listR}%
253 \list@clear{\actions@listR}%
254 \list@clear{\linesinpar@listR}%
255 \list@clear{\linesonpage@listR}
256 \else
257 \list@clearing@reg
258 \list@clear{\linesinpar@listL}%
259 \list@clear{\linesonpage@listL}%
260 \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).
261 \list@clear{\maxlinesinpar@list}

```

Now get the file and interpret it.

```

262 \get@linelistfile{#1}%
263 \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.

264 \ifledRcol
265   \global\page@num=\m@ne
266   \ifx\actionlines@listR\empty
267     \gdef\next@actionlineR{1000000}%
268   \else
269     \gl@p\actionlines@listR\to\next@actionlineR
270     \gl@p\actions@listR\to\next@actionR
271   \fi
272 \else
273   \global\page@num=\m@ne
274   \ifx\actionlines@list\empty
275     \gdef\next@actionline{1000000}%
276   \else
277     \gl@p\actionlines@list\to\next@actionline
278     \gl@p\actions@list\to\next@action
279   \fi
280 \fi}
281
```

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.

12.4 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
\@l what it does depends on whether right text is being processed.

282 \newcommand{\@l@regR}{%
283   \ifx\l@dchset@num\relax \else
284     \advance\absline@numR \@ne
285     \set@line@action
286     \let\l@dchset@num\relax
287     \advance\absline@numR \m@ne
288     \advance\line@numR \m@ne%    % do we need this?
289   \fi
```

```

290 \advance\absline@numR \@ne
291 \ifx\next@page@numR\relax \else
292   \page@action
293   \let\next@page@numR\relax
294 \fi
295 \ifx\sub@change\relax \else
296   \ifnum\sub@change>\z@
297     \sublines@true
298   \else
299     \sublines@false
300   \fi
301   \sub@action
302   \let\sub@change\relax
303 \fi
304 \ifcase\@lockR
305 \or
306   \@lockR \tw@
307 \or\or
308   \@lockR \z@
309 \fi
310 \ifcase\sub@lockR
311 \or
312   \sub@lockR \tw@
313 \or\or
314   \sub@lockR \z@
315 \fi
316 \ifsublines@
317   \ifnum\sub@lockR<\tw@
318     \advance\subline@numR \@ne
319   \fi
320 \else
321   \ifnum\@lockR<\tw@
322     \advance\line@numR \@ne \subline@numR \z@
323   \fi
324 \fi}
325
326 \renewcommand*{\@l}[2]{%
327   \fix@page{#1}%
328   \ifledRcol
329     \@l@regR
330   \else
331     \@l@reg
332   \fi}
333

```

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

```

\fix@page 334 \newcount\last@page@numR
335   \last@page@numR=-10000
336 \renewcommand*{\fix@page}[1]{%
337   \ifledRcol

```



```

338 \ifnum #1=\last@page@numR
339 \else
340 \ifbypage@R
341 \line@numR \z@ \subline@numR \z@
342 \fi
343 \page@numR=#1\relax
344 \last@page@numR=#1\relax
345 \def\next@page@numR{#1}%
346 \fi
347 \else
348 \ifnum #1=\last@page@num
349 \else
350 \ifbypage@
351 \line@num \z@ \subline@num \z@
352 \fi
353 \page@num=#1\relax
354 \last@page@num=#1\relax
355 \def\next@page@num{#1}%
356 \listcsxadd{normal@page@break}{\the\absline@num}
357 \fi
358 \fi}
359

```

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

```

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

```

```

385         \led@warn@BadAdvancelineLine
386         \line@num \z@
387     \fi
388 \fi
389 \fi
390 \set@line@action}
391

```

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

```

392 \renewcommand*{\@set}[1]{%
393     \ifledRcol
394         \ifsublines@
395             \subline@numR=#1\relax
396         \else
397             \line@numR=#1\relax
398         \fi
399     \set@line@action
400 \else
401     \ifsublines@
402         \subline@num=#1\relax
403     \else
404         \line@num=#1\relax
405     \fi
406     \set@line@action
407 \fi}
408

```

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

```

409 \renewcommand*{\l@d@set}[1]{%
410     \ifledRcol
411         \line@numR=#1\relax
412         \advance\line@numR \@ne
413         \def\l@dchset@num{#1}
414     \else
415         \line@num=#1\relax
416         \advance\line@num \@ne
417         \def\l@dchset@num{#1}
418     \fi}
419 \let\l@dchset@num\relax
420

```

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

```

421 \renewcommand*{\page@action}{%
422     \ifledRcol
423         \xright@appenditem{\the\absline@numR}\to\actionlines@listR

```

```

424 \xright@appenditem{\next@page@numR}\to\actions@listR
425 \else
426 \xright@appenditem{\the\absline@num}\to\actionlines@list
427 \xright@appenditem{\next@page@num}\to\actions@list
428 \fi}

```

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

```

429 \renewcommand*{\set@line@action}{%
430 \ifledRcol
431 \xright@appenditem{\the\absline@numR}\to\actionlines@listR
432 \ifsublines@
433 \@l@dttempcnta=-\subline@numR
434 \else
435 \@l@dttempcnta=-\line@numR
436 \fi
437 \advance\@l@dttempcnta by -5000\relax
438 \xright@appenditem{\the\@l@dttempcnta}\to\actions@listR
439 \else
440 \xright@appenditem{\the\absline@num}\to\actionlines@list
441 \ifsublines@
442 \@l@dttempcnta=-\subline@num
443 \else
444 \@l@dttempcnta=-\line@num
445 \fi
446 \advance\@l@dttempcnta by -5000\relax
447 \xright@appenditem{\the\@l@dttempcnta}\to\actions@list
448 \fi}
449

```

`\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.

```

450 \renewcommand*{\sub@action}{%
451 \ifledRcol
452 \xright@appenditem{\the\absline@numR}\to\actionlines@listR
453 \ifsublines@
454 \xright@appenditem{-1001}\to\actions@listR
455 \else
456 \xright@appenditem{-1002}\to\actions@listR
457 \fi
458 \else
459 \xright@appenditem{\the\absline@num}\to\actionlines@list
460 \ifsublines@
461 \xright@appenditem{-1001}\to\actions@list
462 \else
463 \xright@appenditem{-1002}\to\actions@list
464 \fi
465 \fi}
466

```

`\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.

```

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

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

```

```

511     \sub@lockR \z@
512     \fi
513   \else
514     \xright@appenditem{-1004}\to\actions@listR
515     \ifnum\@lockR=\tw@
516       \@lockR \thr@@
517     \else
518       \@lockR \z@
519     \fi
520   \fi}
521
522 \renewcommand*{\do@lockoff}{%
523   \ifledRcol
524     \do@lockoffR
525   \else
526     \do@lockoffL
527   \fi}
528 \global\let\lock@off=\do@lockoff
529

```

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

```

530 \providecommand*{\n@num}{-}
531 \renewcommand*{\n@num}{%
532   \ifledRcol
533     \xright@appenditem{\the\absline@numR}\to\actionlines@listR
534     \xright@appenditem{-1007}\to\actions@listR
535   \else
536     \n@num@reg
537   \fi}
538

```

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

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

```

539   \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.

```

540 \renewcommand*{\@ref}[2]{%
541   \ifledRcol

```

```

542 \global\insert@countR=#1\relax
543 \loop\ifnum\insert@countR>\z@
544 \xright@appenditem{\the\absline@numR}\to\insertlines@listR
545 \global\advance\insert@countR \m@ne
546 \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.

```

547 \begingroup
548 \let\@ref=\dummy@ref
549 \let\page@action=\relax
550 \let\sub@action=\relax
551 \let\set@line@action=\relax
552 \let\@lab=\relax
553 #2
554 \global\endpage@num=\page@numR
555 \global\endline@num=\line@numR
556 \global\endsubline@num=\subline@numR
557 \endgroup

```

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

```

558 \xright@appenditem%
559 {\the\page@numR|\the\line@numR|%
560 \ifsublines@ \the\subline@numR \else 0\fi}%
561 \the\endpage@num|\the\endline@num|%
562 \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.

```

563 #2
564 \else

```

And when not in right text

```

565 \@ref@reg{#1}{#2}%
566 \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.

```

567 \providecommand*\@pend}[1]{%
568 \renewcommand*\@pend}[1]{%
569 \ifbypstart@\global\line@num=0\fi%
570 \xright@appenditem{#1}\to\linesinpar@listL}
571 \providecommand*\@pendR}[1]{%
572 \renewcommand*\@pendR}[1]{%

```

```

573 \ifbypstart@R\global\line@numR=0\fi
574 \xright@appenditem{#1}\to\linesinpar@listR}
575

```

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

```

576 \providecommand*{\@lopL}[1]{%
577 \renewcommand*{\@lopL}[1]{%
578 \xright@appenditem{#1}\to\linesonpage@listL}
579 \providecommand*{\@lopR}[1]{%
580 \renewcommand*{\@lopR}[1]{%
581 \xright@appenditem{#1}\to\linesonpage@listR}
582

```

12.5 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`.

```

583 \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
584 \newif\iffirst@linenum@out@R
585 \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.

```

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

```

```

602      \closeout\linenum@outR%
603      \openout\linenum@outR=#1\relax%
604      \fi
605      \fi}
606

```

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

```

607 \newcommand*{\new@lineL}{%
608   \write\linenum@out{\string\@l[\the\c@page][\thepage]}}

```

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

```

609 \newcommand*{\new@lineR}{%
610   \write\linenum@outR{\string\@r[\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.

```

611 \renewcommand*{\flag@start}{%
612   \ifledRcol
613     \edef\next{\write\linenum@outR{%
614       \string\@ref[\the\insert@countR] []}}%
615     \next
616   \else
617     \edef\next{\write\linenum@out{%
618       \string\@ref[\the\insert@count] []}}%
619     \next
620   \fi}
621 \renewcommand*{\flag@end}{%
622   \ifledRcol
623     \write\linenum@outR{[]}%
624   \else
625     \write\linenum@out{[]}%
626   \fi}

```

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

```

627 \renewcommand*{\startsub}{\dimen0\lastskip
628   \ifdim\dimen0>0pt \unskip \fi
629   \ifledRcol \write\linenum@outR{\string\sub@on}%
630   \else      \write\linenum@out{\string\sub@on}%
631   \fi
632   \ifdim\dimen0>0pt \hskip\dimen0 \fi}
633 \def\endsub{\dimen0\lastskip
634   \ifdim\dimen0>0pt \unskip \fi
635   \ifledRcol \write\linenum@outR{\string\sub@off}%
636   \else      \write\linenum@out{\string\sub@off}%
637   \fi
638   \ifdim\dimen0>0pt \hskip\dimen0 \fi}
639

```


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

```
640 \renewcommand*{\advanceline}[1]{%
641   \ifledRcol \write\linenum@outR{\string\@adv[#1]}%
642   \else      \write\linenum@out{\string\@adv[#1]}%
643   \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.

```
644 \renewcommand*{\setline}[1]{%
645   \ifnum#1<\z@
646     \led@warn@BadSetline
647   \else
648     \ifledRcol \write\linenum@outR{\string\@set[#1]}%
649     \else      \write\linenum@out{\string\@set[#1]}%
650     \fi
651   \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.

```
652 \renewcommand*{\setlinenum}[1]{%
653   \ifnum#1<\z@
654     \led@warn@BadSetlinenum
655   \else
656     \ifledRcol \write\linenum@outR{\string\l@d@set[#1]}
657     \else      \write\linenum@out{\string\l@d@set[#1]} \fi
658   \fi}
659
```

`\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.

```
660 \renewcommand*{\startlock}{%
661   \ifledRcol \write\linenum@outR{\string\lock@on}%
662   \else      \write\linenum@out{\string\lock@on}%
663   \fi}
664 \def\endlock{%
665   \ifledRcol \write\linenum@outR{\string\lock@off}%
666   \else      \write\linenum@out{\string\lock@off}%
667   \fi}
668
```

`\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.

```
669 \renewcommand*{\skipnumbering}{%
670   \ifledRcol \write\linenum@outR{\string\n@num}%
671   \advanceline{-1}%

```

```

672 \else
673   \skipnumbering@reg
674 \fi}
675

```

13 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.

```

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

```

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

```

699 \renewcommand{\edtext}[2]{\leavevmode
700   \begingroup%
701     \renewcommand{\@tag}{\no@expands #1}%
702     \set@line%
703     \ifledRcol \global\insert@countR \z@%
704     \else      \global\insert@count \z@ \fi%
705     \ignorespaces #2\relax%
706     \@ifundefined{xpg@main@language}{%if not polyglossia
707       \flag@start}%
708     {\if@RTL\flag@end\else\flag@start\fi% be careful on the direction of writing with polyglossia
709     }%
710   \endgroup%
711   \showlemma{#1}%
712   \ifx\end@lemmas\empty \else%
713     \gl@p\end@lemmas\to\x@lemma%
714     \x@lemma%
715     \global\let\x@lemma=\relax%
716   \fi%
717   \@ifundefined{xpg@main@language}{%if not polyglossia
718     \flag@end}%
719     {\if@RTL\flag@start\else\flag@end\fi% be careful on the direction of writing with polyglossia
720     }%
721 }
722
```

`\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`.

```

723 \renewcommand*{\set@line}{%
724   \ifledRcol
725     \ifx\line@listR\empty
726       \global\noteschanged@true
727       \xdef\l@d@nums{000|000|000|000|000|000|\edfont@info}%
728     \else
729       \gl@p\line@listR\to\@tempb
730       \xdef\l@d@nums{\@tempb|\edfont@info}%
731       \global\let\@tempb=\undefined
732     \fi
733   \else
734     \ifx\line@list\empty
735       \global\noteschanged@true
736       \xdef\l@d@nums{000|000|000|000|000|000|\edfont@info}%
737     \else
738       \gl@p\line@list\to\@tempb
739       \xdef\l@d@nums{\@tempb|\edfont@info}%
740       \global\let\@tempb=\undefined
741     \fi
742   \fi}
743
```

14 Parallel environments

The initial set up for parallel processing is deceptively simple.

pairs The **pairs** environment is for parallel columns and the **pages** environment for parallel pages.

```
chapterinpages 744 \newenvironment{pairs}{%}
745   \l@dpairingtrue
746   \l@dpagingfalse
747 }{%
748   \l@dpairingfalse
749 }
```

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.

```
750 \newenvironment{pages}{%
751   \let\oldchapter\chapter
752   \let\chapter\chapterinpages
753   \l@dpairingtrue
754   \l@dpagingtrue
755   \setlength{\Lcolwidth}{\textwidth}%
756   \setlength{\Rcolwidth}{\textwidth}%
757 }{%
758   \l@dpairingfalse
759   \l@dpagingfalse
760   \let\chapter\oldchapter
761 }
762 \newcommand{\chapterinpages}{\thispagestyle{plain}%
763   \global\@topnum\z@
764   \@afterindentfalse
765   \secdef\@chapter\@schapter}
766
```

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

```
767 \newif\ifinstanzaL
768 \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.

```
769 \newenvironment{Leftside}{%
770   \ledRcolfalse
771   \let\beginnumbering\beginnumbering\setcounter{pstartL}{1}
772   \let\pstart\pstartL
773   \let\thepstart\thepstartL
```

```

774 \let\pend\pendL
775 \let\memorydump\memorydumpL
776 \Leftsidehook
777 \let\oldstanza\stanza
778 \renewcommand{\stanza}{\oldstanza\global\instanzaLtrue}
779 }{
780 \let\stanza\oldstanza
781 \Leftsidehookend}

```

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

```

\Rightsidehook 782 \newcommand*{\Leftsidehook}{}
\Rightsidehookend 783 \newcommand*{\Leftsidehookend}{}
784 \newcommand*{\Rightsidehook}{}
785 \newcommand*{\Rightsidehookend}{}
786

```

`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.

```

787 \newenvironment{Rightside}{%
788 \ledRcoltrue
789 \let\beginnumbering\beginnumberingR
790 \let\endnumbering\endnumberingR
791 \let\pausenumbering\pausenumberingR
792 \let\resumenumbering\resumenumberingR
793 \let\memorydump\memorydumpR
794 \let\thepstart\thepstartR
795 \let\pstart\pstartR
796 \let\pend\pendR
797 \let\ledpb\ledpbR
798 \let\lednopb\lednopbR
799 \let\lineation\lineationR
800 \Rightsidehook
801 \let\oldstanza\stanza
802 \renewcommand{\stanza}{\oldstanza\global\instanzaRtrue}
803 }{%
804 \ledRcolfalse
805 \let\stanza\oldstanza
806 \Rightsidehookend
807 }
808

```

15 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.

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

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

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

```
809 \newcount\num@linesR
810 \newbox\one@lineR
811 \newcount\par@lineR
```

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

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

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

```
812
813 \newcounter{pstartL}
814 \newcounter{pstartLold}
815 \renewcommand{\thepstartL}{\bfseries\@arabic\c@pstartL}. }
816 \newcounter{pstartR}
817 \newcounter{pstartRold}
818 \renewcommand{\thepstartR}{\bfseries\@arabic\c@pstartR}. }
819
820 \newcommand*{\pstartL}{%
821   \if@nobreak%
822     \let\@oldnobreak\@nobreaktrue%
823   \else%
824     \let\@oldnobreak\@nobreakfalse%
825   \fi%
826   \@nobreaktrue%
827   \ifnumbering \else%
828     \led@err@PstartNotNumbered%
```

```

829 \beginnumbering%
830 \fi%
831 \ifnumberedpar%
832 \led@err@PstartInPstart%
833 \pend%
834 \fi%

```

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

```

835 \ifpst@rtedL\else%
836 \setcounter{pstartLold}{\value{pstartL}}%
837 \list@clear{\inserts@list}%
838 \global\let\next@insert=\empty%
839 \global\pst@rtedLtrue%
840 \fi%
841 \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.

```

842 \global\advance\l@dnumpstartsL \@one%
843 \ifnum\l@dnumpstartsL>\l@dc@maxchunks%
844 \led@err@TooManyPstarts%
845 \global\l@dnumpstartsL=\l@dc@maxchunks%
846 \fi%
847 \global\setnamebox{\l@Lcolrawbox\the\l@dnumpstartsL}=\vbox\bgroup%
848 \ifautopar\else%
849 \ifnumberpstart%
850 \ifsidepstartnum%
851 \else%
852 \thepstartL%
853 \fi%
854 \fi%
855 \fi%
856 \hsize=\Lcolwidth%
857 \numberedpar>true%
858 \iflabelpstart\protected@edef\@currentlabel%
859 {\p@pstartL\thepstartL}\fi%
860 }

861 \newcommand*{\pstartR}{%
862 \if@nobreak%
863 \let\@oldnobreak\@nobreaktrue%
864 \else%
865 \let\@oldnobreak\@nobreakfalse%
866 \fi%
867 \@nobreaktrue%
868 \ifnumberingR \else%
869 \led@err@PstartNotNumbered%
870 \beginnumberingR%
871 \fi%
872 \ifnumberedpar%

```

```

873   \led@err@PstartInPstart%
874   \pendR%
875 \fi%
876 \ifpst@rtedR\else%
877   \setcounter{pstartRold}{\value{pstartR}}%
878   \list@clear{\inserts@listR}%
879   \global\let\next@insertR=\empty%
880   \global\pst@rtedRtrue%
881 \fi%
882 \beginngroup\normal@pars%
883 \global\advance\l@dnumpstartsR \one%
884 \ifnum\l@dnumpstartsR>\l@dc@maxchunks%
885   \led@err@TooManyPstarts%
886   \global\l@dnumpstartsR=\l@dc@maxchunks%
887 \fi%
888 \global\setnamebox{\l@drColrawbox\the\l@dnumpstartsR}=\vbox\bgroup%
889   \ifautopar\else%
890     \ifnumberpstart%
891       \ifsidepstartnum\else%
892         \thepstartR%
893       \fi%
894     \fi%
895   \fi%
896   \hsize=\Rcolwidth%
897   \numberedpar@true%
898   \iflabelpstart\protected@edef\@currentlabel%
899     {\p@pstartR\thepstartR}\fi%
900 }

```

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

```

901 \newcommand*{\pendL}{\ifnumbering \else%
902   \led@err@PendNotNumbered%
903 \fi%
904 \ifnumberedpar@ \else%
905   \led@err@PendNoPstart%
906 \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.

```

907 \l@dzeropenalties%
908 \endgraf\global\num@lines=\prevgraf\egroup%
909 \global\par@line=0%

```

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

```

910 \endgroup%
911 \ignorespaces%
912 \@oldnbreak%

```



```

913 \ifnumberpstart%
914   \addtocounter{pstartL}{1}%
915 \fi
916 \parledgroup@beforenotes@save{L}%
917 }
918

```

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

```

919 \newcommand*{\pendR}{\ifnumberingR \else%
920   \led@err@PendNotNumbered%
921 \fi%
922 \ifnumberedpar@ \else%
923   \led@err@PendNoPstart%
924 \fi%
925 \l@dzeropenalties%
926 \endgraf\global\num@linesR=\prevgraf\egroup%
927 \global\par@lineR=0%
928 \endgroup%
929 \ignorespaces%
930 \oldnobreak%
931 \ifnumberpstart%
932   \addtocounter{pstartR}{1}%
933 \fi%
934 \parledgroup@beforenotes@save{R}%
935 }
936

```

15.2 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.

```

937 \newbox\l@dleftbox
938 \newbox\l@drightbox
939

```

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

```

\countRline 940 \newcount\countLline
941   \countLline \z@
942 \newcount\countRline
943   \countRline \z@
944

```

`\@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`

```

945 \newcount\@donereallinesL
946 \newcount\@donetotallinesL
947 \newcount\@donereallinesR
948 \newcount\@donetotallinesR
949

```

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

```

950 \newcommand*{\do@lineL}{%
951   \advance\countLline \@ne
952   \ifvbox\namebox{\l@dLcolrawbox\the\l@dpscL}%
953   {\vbadness=10000
954     \splittopskip=\z@
955     \do@lineLhook
956     \l@emptyd@ta
957     \global\setbox\one@line=\vsplit\namebox{\l@dLcolrawbox\the\l@dpscL}
958       to\baselineskip}%
959   \IfStrEq{\splitfirstmarks\parledgroup@}{begin}{\parledgroup@notes@startL}{%
960     \unvbox\one@line \global\setbox\one@line=\lastbox
961     \getline@numL
962     \ifnum\@lock>\@ne\inserthangingsymboltrue\else\inserthangingsymbolfalse\fi
963     \setbox\l@dleftbox
964     \hb@xt@ \Lcolwidth{%
965       \affixpstart@numL
966       \affixline@num
967       \l@dld@ta
968       \add@inserts
969       \affixside@note
970       \l@dlsn@te
971       {\ledllfill\hb@xt@ \wd\one@line{\do@insidelineLhook\inserthangingsymbolL\new@lineL\l@
972         \l@drsn@te
973       }}%
974     \add@penaltiesL
975     \global\advance\@donereallinesL\@ne
976     \global\advance\@donetotallinesL\@ne
977   \else
978     \setbox\l@dleftbox \hb@xt@ \Lcolwidth{\hspace*\Lcolwidth}%
979     \global\advance\@donetotallinesL\@ne
980   \fi}
981
982

```

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

```

\do@lineRhook 983 \newcommand*{\do@lineLhook}{}
\do@insidelineLhook 984 \newcommand*{\do@lineRhook}{}
\do@insidelineRhook 985 \newcommand*{\do@insidelineLhook}{}
986 \newcommand*{\do@insidelineRhook}{}
987

```

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

```

988 \newcommand*{\do@lineR}{%
989   \advance\countRline \@ne
990   \ifvbox\namebox{\l@dRcolrawbox\the\l@dpscR}%
991   {\vbadness=10000
992    \splittopskip=\z@
993    \do@lineRhook
994    \l@demptyd@ta
995    \global\setbox\one@lineR=\vsplit\namebox{\l@dRcolrawbox\the\l@dpscR}
996      to\baselineskip}%
997    \IfStrEq{\splitfirstmarks\parledgroup@}{\begin}{\parledgroup@notes@startR}{}
998    \unvbox\one@lineR \global\setbox\one@lineR=\lastbox
999    \getline@numR
1000 \ifnum\@lockR>\@ne\inserthangingsymbolRtrue\else\inserthangingsymbolRfalse\fi
1001   \setbox\l@drightbox
1002   \hb@xt@ \Rcolwidth{%
1003     \affixpstart@numR
1004     \affixline@numR
1005     \l@dld@ta
1006     \add@insertsR
1007     \affixside@noteR
1008     \l@dlsn@te
1009     {\correctchangingR\ledllfill\hb@xt@ \wd\one@lineR{\do@insidelineRhook\inserthangingsymbolR\new@lin
1010       \l@drsn@te
1011   }}%
1012   \add@penaltiesR
1013   \global\advance\@donereallinesR\@ne
1014   \global\advance\@donetotallinesR\@ne
1015 \else
1016   \setbox\l@drightbox \hb@xt@ \Rcolwidth{\hspace*{\Rcolwidth}}
1017   \global\advance\@donetotallinesR\@ne
1018 \fi}
1019
1020
```

15.3 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.

```

1021 \newcommand*{\getline@numR}{%
1022   \global\advance\absline@numR \@ne
1023   \do@actionsR
1024   \do@ballastR
1025   \ifledgroupnotesR\else\ifnumberline
1026     \ifsublines@
1027     \ifnum\sub@lockR<\tw@
1028       \global\advance\subline@numR \@ne
1029     \fi

```

```

1030 \else
1031   \ifnum\@lockR<\tw@
1032     \global\advance\line@numR \@ne
1033     \global\subline@numR \z@
1034   \fi
1035 \fi
1036 \fi
1037 \fi
1038 }
1039 \newcommand*\getline@numL{%
1040   \global\advance\absline@num \@ne
1041   \do@actions
1042   \do@ballast
1043   \iflabeledgroupnotesL@\else\ifnumberline
1044     \ifsublines@
1045       \ifnum\sub@lock<\tw@
1046         \global\advance\subline@num \@ne
1047       \fi
1048     \else
1049       \ifnum\@lock<\tw@
1050         \global\advance\line@num \@ne
1051         \global\subline@num \z@
1052       \fi
1053     \fi
1054 \fi
1055 \fi
1056 }
1057
1058

```

`\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.

```

1059 \newcommand*\do@ballastR{\global\ballast@count=\z@
1060   \begingroup
1061     \advance\absline@numR \@ne
1062     \ifnum\next@actionlineR=\absline@numR
1063       \ifnum\next@actionR>-1001
1064         \global\advance\ballast@count by -\c@ballast
1065       \fi
1066     \fi
1067   \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.

```

1068 \newcommand*\do@actions@fixedcodeR{%
1069   \ifcase\@l@dttempcnta%

```

```

1070 \or% % 1001
1071 \global\sublines@true
1072 \or% % 1002
1073 \global\sublines@false
1074 \or% % 1003
1075 \global\@lockR=\@ne
1076 \or% % 1004
1077 \ifnum\@lockR=\tw@
1078 \global\@lockR=\thr@@
1079 \else
1080 \global\@lockR=\z@
1081 \fi
1082 \or% % 1005
1083 \global\sub@lockR=\@ne
1084 \or% % 1006
1085 \ifnum\sub@lockR=\tw@
1086 \global\sub@lockR=\thr@@
1087 \else
1088 \global\sub@lockR=\z@
1089 \fi
1090 \or% % 1007
1091 \l@dskipnumbertrue
1092 \else
1093 \led@warn@BadAction
1094 \fi}
1095
1096
1097 \newcommand*{\do@actionsR}{%
1098 \global\let\do@actions@nextR=\relax
1099 \@l@dttempcntb=\absline@numR
1100 \ifnum\@l@dttempcntb<\next@actionlineR\else
1101 \ifnum\next@actionR>-1001\relax
1102 \global\page@numR=\next@actionR
1103 \ifbypage@R
1104 \global\line@numR \z@ \global\subline@numR \z@
1105 \fi
1106 \else
1107 \ifnum\next@actionR<-4999\relax % 9/05 added relax here
1108 \@l@dttempcnta=-\next@actionR
1109 \advance\@l@dttempcnta by -5001\relax
1110 \ifsublines@
1111 \global\subline@numR=\@l@dttempcnta
1112 \else
1113 \global\line@numR=\@l@dttempcnta
1114 \fi
1115 \else
1116 \@l@dttempcnta=-\next@actionR
1117 \advance\@l@dttempcnta by -1000\relax
1118 \do@actions@fixedcodeR
1119 \fi

```

```

1120 \fi
1121 \ifx\actionlines@listR\empty
1122 \gdef\next@actionlineR{1000000}%
1123 \else
1124 \glp\actionlines@listR\to\next@actionlineR
1125 \glp\actions@listR\to\next@actionR
1126 \global\let\do@actions@nextR=\do@actionsR
1127 \fi
1128 \fi
1129 \do@actions@nextR}
1130

```

15.4 Line number printing

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

```

\ch@cksub@l@ckR 1131
\ch@ck@l@ckR 1132 \providecommand*\l@dcalcnun}[3]{%
\fx@l@cksR 1133 \ifnum #1 > #2\relax
\affixline@numR 1134 \@l@tempcnta = #1\relax
1135 \advance\@l@tempcnta by -#2\relax
1136 \divide\@l@tempcnta by #3\relax
1137 \multiply\@l@tempcnta by #3\relax
1138 \advance\@l@tempcnta by #2\relax
1139 \else
1140 \@l@tempcnta=#2\relax
1141 \fi}
1142
1143 \newcommand*\ch@cksub@l@ckR}{%
1144 \ifcase\sub@lockR
1145 \or
1146 \ifnum\sublock@disp=\@ne
1147 \@l@tempcntb \z@ \@l@tempcnta \@ne
1148 \fi
1149 \or
1150 \ifnum\sublock@disp=\tw@
1151 \else
1152 \@l@tempcntb \z@ \@l@tempcnta \@ne
1153 \fi
1154 \or
1155 \ifnum\sublock@disp=\z@
1156 \@l@tempcntb \z@ \@l@tempcnta \@ne
1157 \fi
1158 \fi}
1159
1160 \newcommand*\ch@ck@l@ckR}{%
1161 \ifcase\@lockR
1162 \or
1163 \ifnum\lock@disp=\@ne
1164 \@l@tempcntb \z@ \@l@tempcnta \@ne

```

```

1165 \fi
1166 \or
1167 \ifnum\lock@disp=\tw@
1168 \else
1169 \l@dttempcntb \z@ \l@dttempcnta \@ne
1170 \fi
1171 \or
1172 \ifnum\lock@disp=\z@
1173 \l@dttempcntb \z@ \l@dttempcnta \@ne
1174 \fi
1175 \fi}
1176
1177 \newcommand*{\f@x@l@cksR}{%
1178 \ifcase\@lockR
1179 \or
1180 \global\@lockR \tw@
1181 \or \or
1182 \global\@lockR \z@
1183 \fi
1184 \ifcase\sub@lockR
1185 \or
1186 \global\sub@lockR \tw@
1187 \or \or
1188 \global\sub@lockR \z@
1189 \fi}
1190
1191
1192 \newcommand*{\affixline@numR}{%
1193 \ifledgroupnotesR\else\ifnumberline
1194 \ifl@dskipnumber
1195 \global\l@dskipnumberfalse
1196 \else
1197 \ifsublines@
1198 \l@dttempcntb=\subline@numR
1199 \l@dcalcnum{\subline@numR}{\c@firstsublinenumR}{\c@sublinenumincrementR}%
1200 \ch@cksub@lockR
1201 \else
1202 \l@dttempcntb=\line@numR
1203 \ifx\linenumberlist\empty
1204 \l@dcalcnum{\line@numR}{\c@firstlinenumR}{\c@linenumincrementR}%
1205 \else
1206 \l@dttempcnta=\line@numR
1207 \edef\rem@inder{\linenumberlist,\number\line@numR,}%
1208 \edef\sc@n@list{\def\noexpand\sc@n@list
1209 ###1,\number\l@dttempcnta,###2|{\def\noexpand\rem@inder{###2}}}%
1210 \sc@n@list\expandafter\sc@n@list\rem@inder|
1211 \ifx\rem@inder\empty\advance\l@dttempcnta\@ne\fi
1212 \fi
1213 \ch@ck@l@ckR
1214 \fi

```

```

1215 \ifnum\@l@tempcnta=\@l@tempcntb
1216 \if@twocolumn
1217 \if@firstcolumn
1218 \gdef\l@dld@ta{\llap{\leftlinenumR}}}%
1219 \else
1220 \gdef\l@drd@ta{\rlap{\rightlinenumR}}}%
1221 \fi
1222 \else
1223 \l@tempcntb=\line@marginR
1224 \ifnum\@l@tempcntb>\@ne
1225 \advance\@l@tempcntb by\page@numR
1226 \fi
1227 \ifodd\@l@tempcntb
1228 \gdef\l@drd@ta{\rlap{\rightlinenumR}}}%
1229 \else
1230 \gdef\l@dld@ta{\llap{\leftlinenumR}}}%
1231 \fi
1232 \fi
1233 \fi
1234 \f@x@l@cksR
1235 \fi
1236 \fi
1237 \fi}

```

15.5 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 1238
\leftpstartnumR 1239 \newcommand*{\affixpstart@numL}{%
\rightpstartnumR 1240 \ifsidepstartnum
\leftpstartnumL 1241 \if@twocolumn
\rightpstartnumL 1242 \if@firstcolumn
\ifpstartnumR 1243 \gdef\l@dld@ta{\llap{\leftpstartnumL}}}%
1244 \else
1245 \gdef\l@drd@ta{\rlap{\rightpstartnumL}}}%
1246 \fi
1247 \else
1248 \l@tempcntb=\line@margin
1249 \ifnum\@l@tempcntb>\@ne
1250 \advance\@l@tempcntb \page@num
1251 \fi
1252 \ifodd\@l@tempcntb

```



```

1253     \gdef\l@drd@ta{\rlap{\rightpstartnumL}}}%
1254     \else
1255     \gdef\l@dld@ta{\llap{\leftpstartnumL}}}%
1256     \fi
1257     \fi
1258 \fi
1259 }
1260 \newcommand*{\affixpstart@numR}{%
1261 \ifsidepstartnum
1262 \if@twocolumn
1263     \if@firstcolumn
1264     \gdef\l@dld@ta{\llap{\leftpstartnumR}}}%
1265     \else
1266     \gdef\l@drd@ta{\rlap{\rightpstartnumR}}}%
1267     \fi
1268     \else
1269     \@l@tempcntb=\line@marginR
1270     \ifnum\@l@tempcntb>\@ne
1271     \advance\@l@tempcntb \page@numR
1272     \fi
1273     \ifodd\@l@tempcntb
1274     \gdef\l@drd@ta{\rlap{\rightpstartnumR}}}%
1275     \else
1276     \gdef\l@dld@ta{\llap{\leftpstartnumR}}}%
1277     \fi
1278     \fi
1279 \fi
1280 }
1281
1282 \newcommand*{\leftpstartnumL}{
1283 \ifpstartnum
1284 \thepstartL
1285 \kern\linenumsep\global\pstartnumfalse\fi
1286 }
1287 \newcommand*{\rightpstartnumL}{
1288 \ifpstartnum\kern\linenumsep
1289 \thepstartL
1290 \global\pstartnumfalse\fi
1291 }
1292 \newif\ifpstartnumR
1293 \pstartnumRtrue
1294 \newcommand*{\leftpstartnumR}{
1295 \ifpstartnumR
1296 \thepstartR
1297 \kern\linenumsep\global\pstartnumRfalse\fi
1298 }
1299 \newcommand*{\rightpstartnumR}{
1300 \ifpstartnumR\kern\linenumsep
1301 \thepstartR
1302 \global\pstartnumRfalse\fi

```

1303 }

15.6 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.

1304 `\list@create{\inserts@listR}`

`\add@insertsR` The right text version.

```

\add@inserts@nextR 1305 \newcommand*{\add@insertsR}{%
1306   \global\let\add@inserts@nextR=\relax
1307   \ifx\inserts@listR\empty \else
1308     \ifx\next@insertR\empty
1309       \ifx\insertlines@listR\empty
1310         \global\noteschanged@true
1311         \gdef\next@insertR{100000}%
1312       \else
1313         \glp\insertlines@listR\to\next@insertR
1314       \fi
1315     \fi
1316     \ifnum\next@insertR=\absline@numR
1317       \glp\inserts@listR\to\@insertR
1318     \@insertR
1319     \global\let\@insertR=\undefined
1320     \global\let\next@insertR=\empty
1321     \global\let\add@inserts@nextR=\add@insertsR
1322   \fi
1323 \fi
1324 \add@inserts@nextR}
1325
```

15.7 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@dttempcnta` 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@dttempcnta=\ballast@count
\ifnum\num@linesR>\@ne
\global\advance\par@lineR \@ne
```

```

\ifnum\par@lineR=\@ne
  \advance\@l@dttempcnta by \clubpenalty
\fi
\@l@dttempcntb=\par@lineR \advance\@l@dttempcntb \@ne
\ifnum\@l@dttempcntb=\num@linesR
  \advance\@l@dttempcnta by \widowpenalty
\fi
\ifnum\par@lineR<\num@linesR
  \advance\@l@dttempcnta by \interlinepenalty
\fi
\fi
\ifnum\@l@dttempcnta=\z@
  \relax
\else
  \ifnum\@l@dttempcnta>-10000
    \penalty\@l@dttempcnta
  \else
    \penalty -10000
  \fi
\fi}

```

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

```

1326 \newcommand*\add@penaltiesL{}\fi
1327 \newcommand*\add@penaltiesR{}\fi
1328

```

15.8 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.

```

1329 \newcommand*\flush@notesR{}\fi
1330 \xloop
1331   \ifx\inserts@listR\empty \else
1332     \gl@p\inserts@listR\to\@insertR
1333     \@insertR
1334     \global\let\@insertR=\undefined
1335   \repeat}
1336

```

16 Footnotes

16.1 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

```

1337 \def\printlinesR#1|#2|#3|#4|#5|#6|#7|{\begingroup
1338   \setprintlines{#1}{#2}{#3}{#4}{#5}{#6}%
1339   \ifl@d@pnum #1\fullstop\fi
1340   \ifledplinenum \linenumr@p{#2}\Rlineflag\else \symlinenum\fi
1341   \ifl@d@ssub \fullstop \sublinenumr@p{#3}\fi
1342   \ifl@d@dash \endashchar\fi
1343   \ifl@d@pnum #4\fullstop\fi
1344   \ifl@d@elin \linenumr@p{#5}\Rlineflag\fi
1345   \ifl@d@esl \ifl@d@elin \fullstop\fi \sublinenumr@p{#6}\fi
1346 \endgroup}
1347
1348 \let\ledsavedprintlines\printlines
1349
```

17 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.

```

1350 \list@create{\labelref@listR}
1351
```

`\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.

```

1352 \renewcommand*{\edlabel}[1]{\@bsphack
1353   \ifledRcol
1354     \write\linenum@outR{\string\@lab}%
1355     \ifx\labelref@listR\empty
1356       \xdef\label@refs{\zz@@@}%
1357     \else
```

```

1358     \gl@p\labelref@listR\to\label@refs
1359     \fi
1360     \ifvmode
1361         \advancelabel@refs
1362     \fi
1363     \protected@write\@auxout{}%
1364         {\string\l@dmake@labelsR\space\thepage|\label@refs|\the\c@pstartR|{#1}}%
1365 \else
1366     \write\linenum@out{\string\@lab}%
1367     \ifx\labelref@list\empty
1368         \xdef\label@refs{\zz@@@}%
1369     \else
1370         \gl@p\labelref@list\to\label@refs
1371     \fi
1372     \ifvmode
1373         \advancelabel@refs
1374     \fi
1375     \protected@write\@auxout{}%
1376         {\string\l@dmake@labelsR\space\thepage|\label@refs|\the\c@pstartR|{#1}}%
1377 \fi
1378 \@esphack}
1379
1380

```

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

```

1381 \def\l@dmake@labelsR#1|#2|#3|#4|#5{%
1382     \expandafter\ifx\csname the@label#5\endcsname \relax\else
1383         \led@warn@DuplicateLabel{#4}%
1384     \fi
1385     \expandafter\gdef\csname the@label#5\endcsname{#1|#2\Rlineflag|#3|#4}%
1386     \ignorespaces}
1387 \AtBeginDocument{%
1388     \def\l@dmake@labelsR#1|#2|#3|#4|#5{%
1389 }
1390

```

`\@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.

```

1391 \renewcommand*{\@lab}{%
1392     \ifledRcol
1393         \xright@appenditem{\linenumr@p{\line@numR}}|{%
1394             \ifsublines@ \sublinenumr@p{\subline@numR}\else 0\fi}%
1395         \to\labelref@listR
1396     \else
1397         \xright@appenditem{\linenumr@p{\line@num}}|{%
1398             \ifsublines@ \sublinenumr@p{\subline@num}\else 0\fi}%
1399         \to\labelref@list

```

```
1400 \fi}
1401
```

18 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 1402 \newcount\sidenote@marginR
1403 \renewcommand*{\sidenotemargin}[1]{%
1404 \l@dgetsidenote@margin{#1}%
1405 \ifnum\@l@dttempcntb>\m@ne
1406 \ifledRcol
1407 \global\sidenote@marginR=\@l@dttempcntb
1408 \else
1409 \global\sidenote@margin=\@l@dttempcntb
1410 \fi
1411 \fi}}
1412 \sidenotemargin{right}
1413 \global\sidenote@margin=\@ne
1414
```

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

```
\l@dcsnote 1415 \renewcommand*{\l@dlsnote}[1]{%
1416 \begingroup%
1417 \newcommand{\content}{#1}%
1418 \ifnumberedpar@
1419 \ifledRcol%
1420 \xright@appenditem{\noexpand\l@dlsnote{\csexpandonce{content}}}%
1421 \to\inserts@listR
1422 \global\advance\insert@countR \@ne%
1423 \else%
1424 \xright@appenditem{\noexpand\l@dlsnote{\csexpandonce{content}}}%
1425 \to\inserts@list
1426 \global\advance\insert@count \@ne%
1427 \fi
1428 \fi\ignorespaces\endgroup}
1429 \renewcommand*{\l@drsnote}[1]{%
1430 \begingroup%
1431 \newcommand{\content}{#1}%
1432 \ifnumberedpar@
1433 \ifledRcol%
1434 \xright@appenditem{\noexpand\l@drsnote{\csexpandonce{content}}}%
1435 \to\inserts@listR
1436 \global\advance\insert@countR \@ne%
1437 \else%
1438 \xright@appenditem{\noexpand\l@drsnote{\csexpandonce{content}}}%

```

```

1439             \to\inserts@list
1440     \global\advance\insert@count \@ne%
1441     \fi
1442 \fi\ignorespaces\endgroup}
1443 \renewcommand*{\l@dcnote}[1]{%
1444 \begingroup%
1445 \newcommand{\content}{#1}%
1446 \ifnumberedpar@
1447     \ifledRcol%
1448         \xright@appenditem{\noexpand\l@dcnote{\csexpandonce{content}}}%
1449         \to\inserts@listR
1450     \global\advance\insert@countR \@ne%
1451 \else%
1452     \xright@appenditem{\noexpand\l@dcnote{\csexpandonce{content}}}%
1453     \to\inserts@list
1454     \global\advance\insert@count \@ne%
1455 \fi
1456 \fi\ignorespaces\endgroup}
1457

```

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

```

1458 \newcommand*{\affixside@noteR}{%
1459     \def\sidenotecontent@{}%
1460     \numdef{\itemcount@}{0}%
1461     \def\do##1{%
1462         \ifnumequal{\itemcount@}{0}%
1463             {%
1464                 \appto\sidenotecontent@{##1}}% Not print not separator before the 1st note
1465             {\appto\sidenotecontent@{\sidenotesep ##1}%
1466             }%
1467         \numdef{\itemcount@}{\itemcount@+1}%
1468     }%
1469     \dolistloop{\l@dcnotetext}%
1470     \ifnumgreater{\itemcount@}{1}{\eledmac@warning{\itemcount@\space sidenotes on line \the\line@numR}}
1471 \gdef\@templ@d{%
1472 \ifx\@templ@d\l@dcnotetext \else%
1473     \iftwocolumn%
1474         \if@firstcolumn%
1475             \setl@dlp@rbox{\sidenotecontent@}%
1476         \else%
1477             \setl@drp@rbox{\sidenotecontent@}%
1478         \fi%
1479     \else%
1480         \l@dttempcntb=\sidenote@marginR%
1481         \ifnum\l@dttempcntb>\@ne%
1482             \advance\l@dttempcntb by\page@num%
1483         \fi%
1484         \ifodd\l@dttempcntb%
1485             \setl@drp@rbox{\sidenotecontent@t}%
1486         \else%

```

```

1487         \setl@dlp@rbox{\sidenotecontent@}%
1488         \fi%
1489     \fi%
1490 \fi}
1491

```

19 Familiar footnotes

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

```

1492 \renewcommand{\l@dbfnote}[1]{%
1493     \ifnumberedpar@
1494     \gdef\@tag{#1}%
1495     \ifledRcol%
1496         \xright@appenditem{\noexpand\vl@dbfnote{\csexpandonce{\@tag}}{\@thefnmark}}%
1497             \to\inserts@listR
1498         \global\advance\insert@countR \@ne%
1499     \else%
1500         \xright@appenditem{\noexpand\vl@dbfnote{\csexpandonce{\@tag}}{\@thefnmark}}%
1501             \to\inserts@list
1502         \global\advance\insert@count \@ne%
1503     \fi
1504 \fi\ignorespaces}
1505

```

`\normalbfnoteX`

```

1506 \renewcommand{\normalbfnoteX}[2]{%
1507     \ifnumberedpar@
1508     \ifledRcol%
1509     \ifluatex
1510         \footnotelang@lua[R]%
1511     \fi
1512     \@ifundefined{xpg@main@language}%if polyglossia
1513     {}%
1514     {\footnotelang@poly[R]}%
1515     \protected@csxdef{thisfootnote}{\csuse{thefootnote#1}}%
1516     \xright@appenditem{\noexpand\vbfnoteX{#1}{#2}{\csexpandonce{thisfootnote}}}%
1517         \to\inserts@listR
1518     \global\advance\insert@countR \@ne%
1519 \else%
1520     \ifluatex
1521         \footnotelang@lua%
1522     \fi
1523     \@ifundefined{xpg@main@language}%if polyglossia
1524     {}%
1525     {\footnotelang@poly}%
1526     \protected@csxdef{thisfootnote}{\csuse{thefootnote#1}}%
1527     \xright@appenditem{\noexpand\vbfnoteX{#1}{#2}{\csexpandonce{thisfootnote}}}%

```



```

1528             \to\inserts@list
1529     \global\advance\insert@count \@ne%
1530     \fi
1531 \fi\ignorespaces}
1532

```

20 Verse

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

```

\inserthangingsymbolL
\inserthangingsymbolR 1533 \newif\ifinserthangingsymbolR
1534 \newcommand{\inserthangingsymbolL}{%
1535 \ifinserthangingsymbol%
1536     \ifinstanzaL%
1537     \hangingsymbol%
1538     \fi%
1539 \fi}
1540 \newcommand{\inserthangingsymbolR}{%
1541 \ifinserthangingsymbolR%
1542     \ifinstanzaR%
1543     \hangingsymbol%
1544     \fi%
1545 \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 1546 \newcommand{\correcthangingL}{%
1547 \ifl@dpaging\else%
1548     \ifinstanzaL%
1549     \ifinserthangingsymbol%
1550     \hskip \@ifundefined{sza@00}{0}{\expandafter%
1551         \noexpand\csname sza@00@endcsname}\stanzaindentbase%
1552     \fi%
1553     \fi%
1554 \fi}
1555
1556 \newcommand{\correcthangingR}{%
1557 \ifl@dpaging\else%
1558     \ifinstanzaR%
1559     \ifinserthangingsymbolR%
1560     \hskip \@ifundefined{sza@00}{0}{\expandafter%
1561         \noexpand\csname sza@00@endcsname}\stanzaindentbase%
1562     \fi%

```

```

1563     \fi%
1564 \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.

```

1565 \chardef\next=\catcode'\&
1566 \catcode'\&=\active
1567

```

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

```

1568 \newenvironment{astanza}{%
1569   \startstanzahook
1570   \catcode'\&\active
1571   \global\stanza@count\@ne\stanza@modulo\@ne
1572   \ifnum\usernamecount{sza@0@}=\z@
1573     \let\stanza@hang\relax
1574     \let\endlock\relax
1575   \else
1576   %% \interlinepenalty\@M % this screws things up, but I don't know why
1577     \rightskip\z@ plus 1fil\relax
1578   \fi
1579   \ifnum\usernamecount{szp@0@}=\z@
1580     \let\sza@penalty\relax
1581   \fi
1582   \def&{%
1583     \endlock\mbox{}%
1584     \sza@penalty
1585     \global\advance\stanza@count\@ne
1586     \@astanza@line}%
1587   \def\&{%
1588     \endlock\mbox{}
1589     \pend
1590     \endstanzaextra}%
1591   \pstart
1592   \@astanza@line
1593 }{}
1594

```

\@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.

```

1595 \newcommand*{\@astanza@line}{%
1596   \ifnum\value{stanzaindentrepetition}=0
1597     \parindent=\csname sza@\number\stanza@count
1598       @\endcsname\stanzaindentbase
1599   \else
1600     \parindent=\csname sza@\number\stanza@modulo
1601       @\endcsname\stanzaindentbase

```

```

1602      \managestanza@modulo
1603    \fi
1604  \par
1605  \stanza@hang%\mbox{}%
1606  \ignorespaces}
1607

```

Lastly reset the modified category codes.

```

1608  \catcode'\&=\next
1609

```

21 Naming macros

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

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

```

\unhnamebox 1610 \providecommand*\newnamebox}[1]{%
\unvnamebox 1611   \expandafter\newbox\csname #1\endcsname}
\namebox 1612 \providecommand*\setnamebox}[1]{%
  1613   \expandafter\setbox\csname #1\endcsname}
  1614 \providecommand*\unhnamebox}[1]{%
  1615   \expandafter\unhbox\csname #1\endcsname}
  1616 \providecommand*\unvnamebox}[1]{%
  1617   \expandafter\unvbox\csname #1\endcsname}
  1618 \providecommand*\namebox}[1]{%
  1619   \csname #1\endcsname}
  1620

```

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

```

\usenamecount 1621 \providecommand*\newnamecount}[1]{%
  1622   \expandafter\newcount\csname #1\endcsname}
  1623 \providecommand*\usenamecount}[1]{%
  1624   \csname #1\endcsname}
  1625

```

22 Counts and boxes for parallel texts

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

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

```

1626 \newcount\l@dc@maxchunks
1627 \newcommand{\maxchunks}[1]{\l@dc@maxchunks=#1}
1628   \maxchunks{5120}
1629

```

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

```

\l@dnumpstartsR 1630 \newcount\l@dnumpstartsR
1631

```

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

```

\l@psclR 1632 \newcount\l@dpscL
1633 \newcount\l@dpscR
1634

```

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

```

1635 \newcommand*{\l@dsetuprawboxes}{%
1636   \@l@dttempcntb=\l@dc@maxchunks
1637   \loop\ifnum\@l@dttempcntb>\z@
1638     \newnamebox{\l@dLcolrawbox\the\@l@dttempcntb}
1639     \newnamebox{\l@dRcolrawbox\the\@l@dttempcntb}
1640     \advance\@l@dttempcntb \m@ne
1641   \repeat}
1642

```

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

```

1643 \newcommand*{\l@dsetupmaxlinecounts}{%
1644   \@l@dttempcntb=\l@dc@maxchunks
1645   \loop\ifnum\@l@dttempcntb>\z@
1646     \newnamecount{\l@dmaxlinesinpar\the\@l@dttempcntb}
1647     \advance\@l@dttempcntb \m@ne
1648   \repeat}
1649 \newcommand*{\l@dzeromaxlinecounts}{%
1650   \begingroup
1651   \@l@dttempcntb=\l@dc@maxchunks
1652   \loop\ifnum\@l@dttempcntb>\z@
1653     \global\usenamecount{\l@dmaxlinesinpar\the\@l@dttempcntb}=\z@
1654     \advance\@l@dttempcntb \m@ne
1655   \repeat
1656   \endgroup}
1657

```

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

```

1658 \AtBeginDocument{%
1659   \l@dsetuprawboxes

```

```

1660 \l@dssetupmaxlinecounts
1661 \l@dzeromaxlinecounts
1662 \l@dnumpstartsL=\z@
1663 \l@dnumpstartsR=\z@
1664 \l@dpscL=\z@
1665 \l@dpscR=\z@}
1666

```

23 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 1667 \newif\ifl@dusedbabel
\l@dusedbabeltrue 1668 \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 1669 \newif\ifl@dsamelang
1670 \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.
1671 \newcommand*{\l@dchecklang}{%
1672 \l@dsamelangfalse
1673 \edef\@tempa{\theledlanguageL}\edef\@temp{\theledlanguageR}%
1674 \ifx\@tempa\@tempb
1675 \l@dsamelangtrue
1676 \fi}
1677

\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 \.
1678 \newcommand*{\l@dbbl@set@language}[1]{%
1679 \edef\languagename{#1}%

```

```

1680 \select@language{\language}%
1681 \if@filesw
1682   \protected@write\auxout{}\string\select@language{\language}%
1683   \addtocontents{toc}{\string\select@language{\language}%
1684   \addtocontents{lof}{\string\select@language{\language}%
1685   \addtocontents{lot}{\string\select@language{\language}%
1686 \fi}
1687

```

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
1688 \providecommand{\selectlanguage}[1]{%
1689 \newcommand*\l@duselanguage}[1]{%
1690 \gdef\theledlanguageL{%
1691 \gdef\theledlanguageR{%
1692

```

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

```

1693 \AtBeginDocument{%
1694   \ifundefined{xpg@main@language}{%
1695     \ifundefined{bbl@main@language}{%

```

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

```

1696   \l@dusedbabelfalse
1697   \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.

```

1698   \l@dusedbabeltrue
1699   \let\l@doldselectlanguage\selectlanguage
1700   \let\l@doldbbl@set@language\bbl@set@language
1701   \let\bbl@set@language\l@dbbl@set@language
1702   \renewcommand{\selectlanguage}[1]{%
1703     \l@doldselectlanguage{#1}%
1704     \ifledRcol \gdef\theledlanguageR{#1}%
1705     \else      \gdef\theledlanguageL{#1}%
1706   \fi}

```

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

```

1707   \renewcommand*\l@duselanguage[1]{%
1708     \l@doldselectlanguage{#1}%

```

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

```

1709   \gdef\theledlanguageL{\bbl@main@language}%

```

```

1710 \gdef\theledlanguageR{\bbl@main@language}%
1711 }%
1712 }

  If on Polyglossia
1713 { \apptocmd{\xpg@set@language}{%
1714   \ifledRcol \gdef\theledlanguageR{#1}%
1715   \else      \gdef\theledlanguageL{#1}%
1716   \fi}%
1717   \let\l@duselanguage\xpg@set@language
1718   \gdef\theledlanguageL{\xpg@main@language}%
1719   \gdef\theledlanguageR{\xpg@main@language}%
1720 % \end{macrocode}
1721 % That's it.
1722 % \begin{macrocode}
1723 }}

```

24 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.

```

1724 \newcommand*{\Columns}{%
1725   \setcounter{pstartL}{\value{pstartLold}}
1726   \setcounter{pstartR}{\value{pstartRold}}
1727   \ifnum\l@dnumpstartsL=\l@dnumpstartsR\else
1728     \led@err@BadLeftRightPstarts{\the\l@dnumpstartsL}{\the\l@dnumpstartsR}%
1729   \fi

```

Start a group and zero counters, etc.

```

1730 \begingroup
1731   \l@dzeropenalties
1732   \endgraf\global\num@lines=\prevgraf
1733   \global\num@linesR=\prevgraf
1734   \global\par@line=\z@
1735   \global\par@lineR=\z@
1736   \global\l@dpscL=\z@
1737   \global\l@dpscR=\z@

```

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

```

1738   \check@pstarts
1739   \loop\if@pstarts
1740     \global\pstartnumtrue
1741     \global\pstartnumRtrue

```

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

```

1742   \global\advance\l@dpscL \@ne
1743   \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

```
1744 \checkraw@text
1745 \l@dcchecklang
1746 { \loop\ifaraw@text
```

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

```
1747 \ifl@dsamelang
1748 \do@lineL
1749 \do@lineR
1750 \else
1751 \l@duselanguage{\theledlanguageL}%
1752 \do@lineL
1753 \l@duselanguage{\theledlanguageR}%
1754 \do@lineR
1755 \fi
1756 \hb@xt@ \hsize{%
1757 \hfill \unhbox\l@leftbox
1758 \hfill \columnseparator \hfill
1759 \unhbox\l@rightbox
1760 }%
1761 \checkraw@text
1762 \checkverseL
1763 \checkverseR
1764 \checkpb@columns
1765 \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.

```
1766 \@writelinesinparL
1767 \@writelinesinparR
1768 \check@pstarts
1769 \ifbypstart@
1770 \write\linenum@out{\string\@set[1]}
1771 \resetprevline@
1772 \fi
1773 \ifbypstart@R
1774 \write\linenum@outR{\string\@set[1]}
1775 \resetprevline@
1776 \fi
1777 \addtocounter{pstartL}{1}
1778 \addtocounter{pstartR}{1}
1779 \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.

```
1780 \flush@notes
1781 \flush@notesR
```



```

1782 \endgroup
1783 \global\l@dpscL=\z@
1784 \global\l@dpscR=\z@
1785 \global\l@dnumstartsL=\z@
1786 \global\l@dnumstartsR=\z@
1787 \ignorespaces
1788 \global\instanzaLfalse
1789 \global\instanzaRfalse}
1790

```

`\checkpb@columns` `\checkpb@columns` prevent or make pagebreaking in columns, depending of the use of `\ledpb` or `\lednopb`.

```

1791
1792 \newcommand{\checkpb@columns}{%
1793   \newif\if@pb
1794   \newif\if@nopb
1795   \IfStrEq{\led@pb@setting}{before}{%
1796     \numdef{\next@absline}{\the\absline@num+1}%
1797     \numdef{\next@abslineR}{\the\absline@numR+1}%
1798     \xifinlistcs{\next@absline}{l@prev@pb}{\@pbtrue}{}%
1799     \xifinlistcs{\next@abslineR}{l@prev@pbR}{\@pbtrue}{%
1800     \xifinlistcs{\next@absline}{l@prev@nopb}{\@nopbtrue}{}%
1801     \xifinlistcs{\next@abslineR}{l@prev@nopbR}{\@nopbtrue}{%
1802   }{}
1803   \IfStrEq{\led@pb@setting}{after}{%
1804     \xifinlistcs{\the\absline@num}{l@prev@pb}{\@pbtrue}{}%
1805     \xifinlistcs{\the\absline@numR}{l@prev@pbR}{\@pbtrue}{%
1806     \xifinlistcs{\the\absline@num}{l@prev@nopb}{\@nopbtrue}{}%
1807     \xifinlistcs{\the\absline@numR}{l@prev@nopbR}{\@nopbtrue}{%
1808   }{}
1809   \if@nopb\nopagebreak[4]\enlargethispage{\baselineskip}\fi
1810   \if@pb\pagebreak[4]\fi
1811 }

```

`\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).

```

1812 \newcommand*{\columnseparator}{%
1813   \smash{\rule[-0.2\baselineskip]{\columnrulewidth}{1.05\baselineskip}}}
1814 \newdimen\columnrulewidth
1815 \columnrulewidth=\z@
1816

```

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

```

\@pstartstrue 1817 \newif\if@pstarts
\@pstartsfalse 1818 \newcommand*{\check@pstarts}{%
\check@pstarts 1819   \@pstartsfalse
1820   \ifnum\l@dnumstartsL>\l@dpscL

```

```

1821     \@pstartstrue
1822   \else
1823     \ifnum\l@dnumpstartsR>\l@dpscR
1824       \@pstartstrue
1825     \fi
1826   \fi
1827 }
1828
\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
1829 \newif\ifaraw@text
1830   \araw@textfalse
1831 \newcommand*{\checkraw@text}{%
1832   \araw@textfalse
1833   \ifvbox\namebox{\l@dLcolrawbox\the\l@dpscL}
1834     \araw@texttrue
1835   \else
1836     \ifvbox\namebox{\l@dRcolrawbox\the\l@dpscR}
1837       \araw@texttrue
1838     \fi
1839   \fi
1840 }
1841
\@writelinesinparL These write the number of text lines in a chunk to the section files, and then
\@writelinesinparR afterwards zero the counter.
1842 \newcommand*{\@writelinesinparL}{%
1843   \edef\next{%
1844     \write\linenum@out{\string\@pend[\the\@donereallinesL]}}%
1845   \next
1846   \global\@donereallinesL \z@}
1847 \newcommand*{\@writelinesinparR}{%
1848   \edef\next{%
1849     \write\linenum@outR{\string\@pendR[\the\@donereallinesR]}}%
1850   \next
1851   \global\@donereallinesR \z@}
1852

```

25 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
1853 \newcount\numpagelinesL
1854 \newcount\numpagelinesR

```

```

1855 \newcount\l@dminpagelines
1856

```

\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.

```

1857 \newcommand*{\Pages}{%
1858   \setcounter{pstartL}{\value{pstartLold}}
1859   \setcounter{pstartR}{\value{pstartRold}}
1860   \parledgroup@notespacing@set@correction
1861   \typeout{}
1862   \typeout{***** PAGES *****}
1863   \ifnum\l@dnumpstartsL=\l@dnumpstartsR\else
1864     \led@err@BadLeftRightPstarts{\the\l@dnumpstartsL}{\the\l@dnumpstartsR}%
1865   \fi

```

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

```

1866 \cleartol@evenpage
1867 \begingroup
1868   \l@dzeropenalties
1869   \endgraf\global\num@lines=\prevgraf
1870   \global\num@linesR=\prevgraf
1871   \global\par@line=\z@
1872   \global\par@lineR=\z@
1873   \global\l@dpscL=\z@
1874   \global\l@dpscR=\z@
1875   \writtenlinesLfalse
1876   \writtenlinesRfalse

```

Check if there are chunks to be processed.

```

1877   \check@pstarts
1878   \loop\if@pstarts

```

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

```

1879     \global\advance\l@dpscL \@ne
1880     \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**.

```

1881     \getlinesfromparlistL
1882     \getlinesfromparlistR
1883     \l@dcalc@maxoftwo{\cs@linesinparL}{\cs@linesinparR}%
1884     {\usenamecount{\l@dmaxlinesinpar\the\l@dpscL}}%
1885     \check@pstarts
1886   \repeat

```

Zero the counts again, ready for the next bit.

```

1887   \global\l@dpscL=\z@
1888   \global\l@dpscR=\z@

```

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

```
1889 \getlinesfrompagelistL
1890 \getlinesfrompagelistR
1891 \l@dcalc@minoftwo{\@cs@linesonpageL}{\@cs@linesonpageR}%
1892 {\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.

```
1893 \check@pstarts
1894 \if@pstarts
```

Increment the chunk counts to get the first pair.

```
1895 \global\advance\l@dpscL \@ne
1896 \global\advance\l@dpscR \@ne
```

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

```
1897 \global\@donereallinesL=\z@
1898 \global\@donetotallinesL=\z@
1899 \global\@donereallinesR=\z@
1900 \global\@donetotallinesR=\z@
```

Start a loop over the boxes (chunks).

```
1901 \checkraw@text
1902 % \begingroup
1903 { \loop\ifaraw@text
```

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

```
1904 \checkpageL
1905 \l@duselanguage{\theledlanguageL}%
1906 %% \begingroup
1907 { \loop\ifl@dsamepage
```

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

```
1908 \do@lineL
1909 \advance\l@numpagelinesL \@ne
1910 \ifshiftedpstarts
1911 \ifdim\ht\l@dleftbox>0pt\hb@xt@ \hsize{\ledstrutL\unhbox\l@dleftbox}
1912 \else%
1913 \parledgroup@correction@notespacing{L}
1914 \hb@xt@ \hsize{\ledstrutL\unhbox\l@dleftbox}%
1915 \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.

```
1916 \get@nextboxL
1917 \checkpageL
1918 \checkverseL
1919 \checkpbL
1920 \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.

```

1921      \ifl@dpagfull
1922      \@writelinesonpageL{\the\numpagelinesL}%
1923      \else
1924      \@writelinesonpageL{1000}%
1925      \fi

```

Reset to zero the left-page line count, clear the page to get onto the facing (odd, right) page, and reinitialize the accumulated dimension of interline correction for notes in parallel ledgroup.

```

1926      \numpagelinesL \z@
1927      \parledgroup@correction@notespacing@init
1928      \clearl@dleftpage }%

```

Now do the same for the right text.

```

1929      \checkpageR
1930      \l@duselanguage{\theledlanguageR}%
1931 {
1932      \loop\ifl@dsamepage
1933      \do@lineR
1934      \advance\numpagelinesR \@ne
1935      \ifshiftedpstarts
1936      \ifdim\ht\l@drightbox>0pt\hb@xt@ \hsize{\ledstrutR\unhbox\l@drightbox}\fi%
1937      \else%
1938      \parledgroup@correction@notespacing{R}
1939      \hb@xt@ \hsize{\ledstrutR\unhbox\l@drightbox}%
1940      \fi
1941      \get@nextboxR
1942      \checkpageR
1943      \checkverseR
1944      \checkpbR
1945      \repeat
1946      \ifl@dpagfull
1947      \@writelinesonpageR{\the\numpagelinesR}%
1948      \else
1949      \@writelinesonpageR{1000}%
1950      \fi
1951      \numpagelinesR=\z@
1952      \parledgroup@correction@notespacing@init

```

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

```

1952      \clearl@drighthpage}

```

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

```

1953      \checkraw@text
1954      \ifaraw@text

```

```

1955      \getlinesfrompagelistL
1956      \getlinesfrompagelistR
1957      \l@dcalc@minoftwo{\@cs@linesonpageL}{\@cs@linesonpageR}%
1958      {\l@dminpagelines}%
1959      \fi
1960      \repeat}

```

We have now output the text from all the chunks.

```

1961      \fi
      Make sure that there are no inserts hanging around.
1962      \flush@notes
1963      \flush@notesR
1964      \endgroup

```

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

```

1965      \global\l@dpscL=\z@
1966      \global\l@dpscR=\z@
1967      \global\l@dnumpestartsL=\z@
1968      \global\l@dnumpestartsR=\z@
1969      \global\instanzaLfalse
1970      \global\instanzaRfalse
1971      \ignorespaces}
1972

```

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

```

\ledstrutR 1973 \newcommand*{\ledstrutL}{\strut}
1974 \newcommand*{\ledstrutR}{\strut}
1975

```

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

```

1976 \providecommand{\cleartoevenpage}[1][\@empty]{%
1977   \clearpage
1978   \ifodd\c@page\hbox{#1}\clearpage\fi}
1979 \newcommand*{\cleartol@devenpage}{%
1980   \ifdim\pagetotal<\topskip% on an empty page
1981     \else
1982       \clearpage
1983     \fi
1984     \ifodd\c@page\hbox{} \clearpage\fi}
1985 \newcommand*{\clearl@dleftpage}{%
1986   \clearpage
1987   \ifodd\c@page\else
1988     \led@err@LeftOnRightPage
1989     \hbox{}%

```

```

1990     \cleardoublepage
1991     \fi}
1992 \newcommand*{\clearl@drighthouse}{%
1993     \clearpage
1994     \ifodd\c@page
1995         \led@err@RightOnLeftPage
1996         \hbox{}%
1997     \cleartoevenpage
1998     \fi}
1999

```

`\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 2000 \newcommand*{\getlinesfromparlistL}{%
2001     \ifx\linesinpar@listL\empty
2002         \gdef\cs@linesinparL{0}%
2003     \else
2004         \gl@p\linesinpar@listL\to\cs@linesinparL
2005     \fi}
2006 \newcommand*{\getlinesfromparlistR}{%
2007     \ifx\linesinpar@listR\empty
2008         \gdef\cs@linesinparR{0}%
2009     \else
2010         \gl@p\linesinpar@listR\to\cs@linesinparR
2011     \fi}
2012

```

`\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 2013 \newcommand*{\getlinesfrompagelistL}{%
2014     \ifx\linesonpage@listL\empty
2015         \gdef\cs@linesonpageL{1000}%
2016     \else
2017         \gl@p\linesonpage@listL\to\cs@linesonpageL
2018     \fi}
2019 \newcommand*{\getlinesfrompagelistR}{%
2020     \ifx\linesonpage@listR\empty
2021         \gdef\cs@linesonpageR{1000}%
2022     \else
2023         \gl@p\linesonpage@listR\to\cs@linesonpageR
2024     \fi}
2025

```

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

```

2026 \newcommand*{\@writelinesonpageL}[1]{%
2027     \edef\next{\write\linenumout{\string\@lopL{#1}}}%

```

```

2028 \next}
2029 \newcommand*{\@writelinesonpageR}[1]{%
2030 \edef\next{\write\linenum@outR{\string\@lopR{#1}}}%
2031 \next}
2032

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

2033 \newcommand*{\l@dcalc@maxoftwo}[3]{%
2034 \ifnum #2>#1\relax
2035     #3=#2\relax
2036 \else
2037     #3=#1\relax
2038 \fi}
2039 \newcommand*{\l@dcalc@minoftwo}[3]{%
2040 \ifnum #2<#1\relax
2041     #3=#2\relax
2042 \else
2043     #3=#1\relax
2044 \fi}
2045

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

\checkpageL 2046 \newif\ifl@dsamepage
\checkpageR 2047 \l@dsamepagetrue
2048 \newif\ifl@dpagfull
2049
2050 \newcommand*{\checkpageL}{%
2051 \l@dpagfulltrue
2052 \l@dsamepagetrue
2053 \check@goal
2054 \ifdim\pagetotal<\ledthegoal
2055 \ifnum\numpagelinesL<\l@dminpagelines
2056 \else
2057 \l@dsamepagefalse
2058 \l@dpagfullfalse
2059 \fi
2060 \else
2061 \l@dsamepagefalse
2062 \l@dpagfulltrue
2063 \fi}
2064 \newcommand*{\checkpageR}{%

```



```

2065 \l@dpagfulltrue
2066 \l@dsamepagetrue
2067 \check@goal
2068 \ifdim\pagetotal<\ledthegoal
2069 \ifnum\numpagelinesR<\l@dmnpagelines
2070 \else
2071 \l@dsamepagefalse
2072 \l@dpagfullfalse
2073 \fi
2074 \else
2075 \l@dsamepagefalse
2076 \l@dpagfulltrue
2077 \fi}
2078

```

`\checkpbL` `\checkpbL` and `\checkpbR` are called after each line is printed, and after the page is checked. These commands correct page breaks depending on `\ledpb` and `\lednopb`.

```

2079 \newcommand{\checkpbL}{
2080 \IfStrEq{\led@pb@setting}{after}{
2081 \xifinlistcs{\the\absline@num}{\l@prev@pb}{\l@dpagfulltrue\l@dsamepagefalse}{
2082 \xifinlistcs{\the\absline@num}{\l@prev@nopb}{\l@dpagfullfalse\l@dsamepagetrue}{
2083 }}
2084 \IfStrEq{\led@pb@setting}{before}{
2085 \numdef{\next@absline}{\the\absline@num+1}
2086 \xifinlistcs{\next@absline}{\l@prev@pb}{\l@dpagfulltrue\l@dsamepagefalse}{
2087 \xifinlistcs{\next@absline}{\l@prev@nopb}{\l@dpagfullfalse\l@dsamepagetrue}{
2088 }}
2089 }
2090
2091 \newcommand{\checkpbR}{
2092 \IfStrEq{\led@pb@setting}{after}{
2093 \xifinlistcs{\the\absline@numR}{\l@prev@pbR}{\l@dpagfulltrue\l@dsamepagefalse}{
2094 \xifinlistcs{\the\absline@numR}{\l@prev@nopbR}{\l@dpagfullfalse\l@dsamepagetrue}{
2095 }}
2096 \IfStrEq{\led@pb@setting}{before}{
2097 \numdef{\next@abslineR}{\the\absline@numR+1}
2098 \xifinlistcs{\next@abslineR}{\l@prev@pbR}{\l@dpagfulltrue\l@dsamepagefalse}{
2099 \xifinlistcs{\next@abslineR}{\l@prev@nopbR}{\l@dpagfullfalse\l@dsamepagetrue}{
2100 }}
2101 }

```

`\checkverseL` `\checkverseL` and `\checkverseR` are called after each line is printed. They prevent page break inside verse.

```

2102 \newcommand{\checkverseL}{
2103 \ifinstanzaL
2104 \iflednopbinverse
2105 \ifinserthangingsymbol
2106 \numgdef{\prev@abslineverse}{\the\absline@num-1}

```

```

2107     \IfStrEq{\led@pb@setting}{after}{\lednopbnum{\prev@abslineverse}}{}
2108     \IfStrEq{\led@pb@setting}{before}{\ifnum\numpagelinesL<3\ledpbnum{\prev@abslineverse}
2109     \fi
2110     \fi
2111 \fi
2112 }
2113 \newcommand{\checkverseR}{
2114 \ifinstanzaR
2115     \iflednopbinverse
2116         \ifinserthangingsymbolR
2117             \numgdef{\prev@abslineverse}{\the\absline@numR-1}
2118             \IfStrEq{\led@pb@setting}{after}{\lednopbnumR{\prev@abslineverse}}{}
2119             \IfStrEq{\led@pb@setting}{before}{\ifnum\numpagelinesR<3\ledpbnumR{\prev@abslineverse}
2120             \fi
2121             \fi
2122 \fi
2123 }

```

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

```

2124 \newdimen\ledthegoal
2125 \ifshiftedpstarts
2126     \newcommand*{\goalfraction}{0.95}
2127 \else
2128     \newcommand*{\goalfraction}{0.9}
2129 \fi
2130
2131 \newcommand*{\check@goal}{%
2132     \ledthegoal=\goalfraction\pagegoal}
2133

```

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

```

\ifwrittenlinesL 2134 \newif\ifwrittenlinesL
2135 \newif\ifwrittenlinesR
2136

```

`\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.

```

2137 \newcommand*{\get@nextboxL}{%
2138     \ifvbox\namebox{1@dLcolrawbox\the\1@dpscl}% box is not empty

    The current box is not empty; do nothing.

2139     \else%
                                                box is empty

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

2140     \ifnum\usernamecount{1@dmaxlinesinpar\the\1@dpscl}>\@donetotallinesL
2141     \parledgroup@notes@endL
2142     \else

```

Sufficient lines have been output.

```

2143      \ifnum\usernamecount{l@dmxlinesinpar\the\l@dpscL}=\@donetotallinesL
2144      \parledgroup@notes@endL
2145      \fi
2146      \ifwrittenlinesL
2147      \else

```

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

```

2148      \@writelinesinparL
2149      \writtenlinesLtrue
2150      \fi
2151      \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.

```

2152      \writtenlinesLfalse
2153      \ifbypstart@
2154      \ifnum\value{pstartL}<\value{pstartLold}
2155      \else
2156      \global\line@num=0
2157      \resetprevline@
2158      \fi
2159      \fi
2160      \addtocounter{pstartL}{1}
2161      \global\pstartnumtrue
2162      \l@dcalc@maxoftwo{\the\usernamecount{l@dmxlinesinpar\the\l@dpscL}}%
2163      {\the\@donetotallinesL}%
2164      {\usernamecount{l@dmxlinesinpar\the\l@dpscL}}%
2165      \global\@donetotallinesL \z@
2166      \global\advance\l@dpscL \@ne
2167      \parledgroup@notes@endL
2168      \parledgroup@correction@notespadding@final{L}
2169      \fi
2170      \fi
2171      \fi}

2172 \newcommand*{\get@nextboxR}{%
2173   \ifvbox\namebox{l@dRcolrawbox\the\l@dpscR}% box is not empty
2174   \else% box is empty
2175     \ifnum\usernamecount{l@dmxlinesinpar\the\l@dpscR}>\@donetotallinesR
2176     \parledgroup@notes@endR
2177     \else
2178       \ifnum\usernamecount{l@dmxlinesinpar\the\l@dpscR}=\@donetotallinesR
2179       \parledgroup@notes@endR
2180       \fi
2181       \ifwrittenlinesR
2182       \else
2183       \@writelinesinparR
2184       \writtenlinesRtrue
2185       \fi

```

```

2186 \ifnum\l@dnumstartR>\l@dpscR
2187 \writtenlinesRfalse
2188 \ifbypstartR
2189 \ifnum\value{pstartR}<\value{pstartRold}
2190 \else
2191 \global\line@numR=0
2192 \resetprevline@
2193 \fi
2194 \fi
2195 \addtocounter{pstartR}{1}
2196 \global\pstartnumRtrue
2197 \l@dcalc@maxoftwo{\the\usernamecount{1@dmaxlinesinpar\the\l@dpscR}}%
2198 {\the\@donetotallinesR}%
2199 {\usernamecount{1@dmaxlinesinpar\the\l@dpscR}}%
2200 \global\@donetotallinesR \z@
2201 \global\advance\l@dpscR \@ne
2202 \parledgroup@notes@endR
2203 \parledgroup@correction@notes@spacing@final{R}
2204 \fi
2205 \fi
2206 \fi}
2207

```

26 Page break/no page break, depending on the specific line

We need to adapt the macro of the homonym section of eledmac to eledpar.

`\prev@pbR` The `\l@prev@pbR` macro is a etoolbox list, which contains the lines in which page breaks occur (before or after). The `\l@prev@nopbR` macro is a etoolbox list, which contains the lines in which NO page breaks occur (before or after).

```

2208 \def\l@prev@pbR{}
2209 \def\l@prev@nopbR{}

```

`\ledpbR` The `\ledpbR` macro writes the call to `\led@pbR` in line-list file. The `\ledpbnumR` macro writes the call to `\led@pbnumR` in line-list file. The `\lednopbR` macro writes the call to `\led@nopbR` in line-list file. The `\lednopbnumR` macro writes the call to `\led@nopbnumR` in line-list file.

```

2210 \newcommand{\ledpbR}{\write\linenum@outR{\string\led@pbR}}
2211 \newcommand{\ledpbnumR}[1]{\write\linenum@outR{\string\led@pbnumR{#1}}}
2212 \newcommand{\lednopbR}{\write\linenum@outR{\string\led@nopbR}}
2213 \newcommand{\lednopbnumR}[1]{\write\linenum@outR{\string\led@nopbnumR{#1}}}

```

`\led@pbR` The `\led@pbR` add the absolute line number in the `\prev@pbR` list. The `\led@pbnumR` add the argument in the `\prev@pbR` list. The `\led@nopbR` add the absolute line number in the `\prev@nopbR` list. The `\led@nopbnumR` add the argument in the `\prev@nopbR` list.

```

2214 \newcommand{\led@pbR}{\listcsxadd{1@prev@pbR}{\the\absline@numR}}
2215 \newcommand{\led@pbnumR}[1]{\listcsxadd{1@prev@pbR}{#1}}
2216 \newcommand{\led@nopbR}{\listcsxadd{1@prev@nopbR}{\the\absline@numR}}
2217 \newcommand{\led@nopbnumR}[1]{\listcsxadd{1@prev@nopbR}{#1}}

```

27 Parallel ledgroup

```

\parledgroup@ The marks \parledgroup contains information about the beginnings and endings
\parledgroupseries@ of notes in a parallel ledgroup. \parledgroupseries contains the footnote series.
\parledgroup@type@ \parledgroupseries contains the type of the footnote: critical (Xfootnote) or
                    familiar (footnoteX).

2218 \newmarks\parledgroup@
2219 \newmarks\parledgroup@series
2220 \newmarks\parledgroup@type

\parledgroup@notes@startL \parledgroup@notes@startL and \parledgroup@notes@startR are used to
\parledgroup@notes@startR mark the beginning of a note series in a parallel ledgroup.

2221 \newcommand{\parledgroup@notes@startL}{%
2222   \ifnum\usenamecount{1@dmaxlinesinpar\the\l@dpscL}>0%
2223     \IfStrEq{\splitfirstmarks\parledgroup@type}{footnoteX}{\csuse{hooknoteX@\splitfirstmarks\parledg
2224       \IfStrEq{\splitfirstmarks\parledgroup@type}{Xfootnote}{\csuse{hookXnote@\splitfirstmarks\parledg
2225     \fi%
2226     \global\ledgroupnotesL@true%
2227     \insert@noterule@ledgroup{L}%
2228 }
2229 \newcommand{\parledgroup@notes@startR}{%
2230   \ifnum\usenamecount{1@dmaxlinesinpar\the\l@dpscR}>0%
2231     \IfStrEq{\splitfirstmarks\parledgroup@type}{footnoteX}{\csuse{hooknoteX@\splitfirstmarks\parledg
2232     \IfStrEq{\splitfirstmarks\parledgroup@type}{Xfootnote}{\csuse{hookXnote@\splitfirstmarks\parledg
2233   \fi%
2234   \global\ledgroupnotesR@true%
2235   \insert@noterule@ledgroup{R}%
2236 }

\parledgroup@notes@endL \parledgroup@notes@endL and \parledgroup@notes@endR are used to mark the
\parledgroup@notes@endR end of a note series in a parallel ledgroup.

2237 \newcommand{\parledgroup@notes@endL}{%
2238   \global\ledgroupnotesL@false%
2239 }
2240 \newcommand{\parledgroup@notes@endR}{%
2241   \global\ledgroupnotesR@false%
2242 }

\insert@noterule@ledgroup A \vskip is not used when the boxes are constructed. So we insert it before
                          ledgroup note series when paralling lines are constructed. This is the goal of
                          \insert@noterule@ledgroup

2243 \newcommand{\insert@noterule@ledgroup}[1]{
2244   \IfStrEq{\splitbotmarks\parledgroup@}{begin}{%

```

```

2245 \IfStrEq{\splitbotmarks\parledgroup@type}{Xfootnote}{
2246 \csuse{ifledgroupnotes#1@}
2247 \vskip\skip\csuse{mp\splitbotmarks\parledgroup@series footins}
2248 \csuse{\splitbotmarks\parledgroup@series footnoterule}
2249 \fi
2250 }
2251 {}
2252 \IfStrEq{\splitbotmarks\parledgroup@type}{footnoteX}{
2253 \csuse{ifledgroupnotes#1@}
2254 \vskip\skip\csuse{mpfootins\splitbotmarks\parledgroup@series}
2255 \csuse{footnoterule\splitbotmarks\parledgroup@series}
2256 \fi
2257 }{}
2258 }
2259 {}
2260 }

```

`\parledgroupnotespacing` `\parledgroupnotespacing` can be redefined by the user to change the interline spacing of ledgroup notes.

```
2261 \newcommand{\parledgroupnotespacing}{}

```

`\parledgroup@notespacing@correction` `\parledgroup@notespacing@correction` is the difference between a normal line skip and a line skip in a note. It's set by `\parledgroup@notespacing@set@correction`, called at the beginning of `\Pages`.

```

2262 \dimdef{\parledgroup@notespacing@correction}{0pt}
2263 \newcommand{\parledgroup@notespacing@set@correction}{%
2264 {\notefontsetup\parledgroupnotespacing\dimgdef{\temp@spacing}{\baselineskip}}%
2265 \dimgdef{\parledgroup@notespacing@correction}{\baselineskip-\temp@spacing}%
2266 }

```

`\parledgroup@correction@notespacing@init` `\parledgroup@correction@notespacing@init` sets the value of accumulated corrections of note spacing to 0 pt. It's called at the beginning of each pages AND at the end of each ledgroup.

```

2267 \newcommand{\parledgroup@correction@notespacing@init}{
2268 \dimdef{\parledgroup@notespacing@correction@accumulated}{0pt}
2269 \dimdef{\parledgroup@notespacing@correction@modulo}{0pt}
2270 }
2271 \parledgroup@correction@notespacing@init

```

`\parledgroup@correction@notespacing@final` `\parledgroup@correction@notespacing@final` adds the total space deleted because of correction for notes, in a parallel ledgroup. It also adds the space needed by the other side spaces between note rules and notes. It's called after the print of each pstart/pend.

```

2272 \newcommand{\parledgroup@correction@notespacing@final}[1]{
2273 \ifparledgroup
2274 \vspace{\parledgroup@notespacing@correction@accumulated}
2275 \parledgroup@correction@notespacing@init%
2276 \ifstrequal{#1}{L}{
2277 \numdef{\@checking}{\the\l@dpscL-1}

```

```

2278   }{
2279       \numdef{\@checking}{\the\l@dpscR-1}
2280   }
2281   \dimdef{\@beforenotes@current@diff}{\csuse{@parledgroup@beforenotes@\@checking L}-\csuse{@parledg
2282   \ifstrequal{#1}{L}%
2283       {% Left
2284       \ifdimgreater{\@beforenotes@current@diff}{0pt}{\vspace{-\@beforenotes@current@diff}}%
2285       }%
2286       {% Right
2287       \ifdimgreater{\@beforenotes@current@diff}{0pt}{\vspace{\@beforenotes@current@diff}}{}}
2288       }%
2289   \fi
2290 }

```

`\parledgroup@correction@notespacing` is used before each printed line. If it's a line of notes in parallel ledgroup, the space `\parledgroup@notespacing@correction` is decreased, to make interline space correct. The decreased space is added to `\parledgroup@notespacing@correction@accumulated` and `\parledgroup@notespacing@correction@modulo`. If `\parledgroup@notespacing@correction@modulo` is equal or greater than `\baselineskip`:

- It is decreased by `\baselineskip`.
- The total of line number in the current page is decreased by one.

For example, suppose an normal interline of 24 pt and interline for note of 12 pt. That means that the two lines of notes take the place of one normal line. For every two lines of notes, the line total for the current place is decreased by one.

```

2291 {}
2292 \newcommand{\parledgroup@correction@notespacing}[1]{%
2293     \csuse{ifledgroupnotes#1@}%
2294     \vspace{-\parledgroup@notespacing@correction}%
2295     \dimdef{\parledgroup@notespacing@correction@accumulated}{\parledgroup@notespacing@correction@ac
2296     \dimdef{\parledgroup@notespacing@correction@modulo}{\parledgroup@notespacing@correction@modulo+
2297     \ifdimless{\parledgroup@notespacing@correction@modulo}{\baselineskip}{\advance\numpagelinesL
2298     \dimdef{\parledgroup@notespacing@correction@modulo}{\parledgroup@notespacing@correction@modulo-
2299     }% mean greater than equal
2300     \fi%
2301 }

```

`\parledgroup@beforenotesL` and `\parledgroup@beforenotesR` store the total of space before notes in the current parallel ledgroup.

```

2302 \dimdef\parledgroup@beforenotesL{0pt}
2303 \dimdef\parledgroup@beforenotesR{0pt}

```

The macro `\parledgroup@beforenotes@save` dumps the space before notes of the current parallel ledgroup in a macro named with the current `pstart` number.

```

2304 \newcommand{\parledgroup@beforenotes@save}[1]{
2305     \ifparledgroup

```

```
2306 \csdimgdef{@parledgroup@beforenotes@the\csuse{1@dnumstarts#1}#1}{\csuse{parledgroup@beforenotes#1}{Opt}}
2307 \csdimgdef{parledgroup@beforenotes#1}{Opt}
2308 \fi
2309 }
```

28 The End

i/code_i

Appendix A Some things to do when changing version

Appendix A.1 Migration to eledpar 1.4.3

Version 1.4.3 corrects a bug added in version 0.12, which made hanging verse automatically flush right, despite the given value of the first element of the `\setstanzaindents` command.

If, however, you want to return to automatic flush-right margins for verses with hanging indents, you have to redefine the `\hangingsymbol` command.

```
\renewcommand{\hangingsymbol}{\protect\hfill}
```

See the two following examples:

With standard `\hangingsymbol`:

A very long verse should be sometime hanged. The position of the hanging verse is fixed.

With the modification of `\hangingsymbol`:

A very long verse should sometimes be hanging. And we can see that an hanging verse is flush right.

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>\@M</code>	1576
<code>\&</code>	1565, 1566, 1570, 1587, 1608	<code>\@adv</code> <u>360</u> , 641, 642

- \@afterindentfalse 764
 - \@arabic 216, 217, 815, 818
 - \@astanza@line 1586, 1592, 1595
 - \@auxout 1363, 1375, 1682
 - \@beforenotes@current@diff
..... 2281, 2284, 2287
 - \@chapter 765
 - \@checking 2277, 2279, 2281
 - \@cs@linesinparL 1883, 2000
 - \@cs@linesinparR 1883, 2000
 - \@cs@linesonpageL .. 1891, 1957, 2013
 - \@cs@linesonpageR .. 1891, 1957, 2013
 - \@currentlabel 858, 898
 - \@donereallinesL
..... 945, 975, 1844, 1846, 1897
 - \@donereallinesR
..... 945, 1013, 1849, 1851, 1899
 - \@donetotallinesL 945, 976,
979, 1898, 2140, 2143, 2163, 2165
 - \@donetotallinesR 945, 1014,
1017, 1900, 2175, 2178, 2198, 2200
 - \@insertR 1317–1319, 1332–1334
 - \@l 282, 608, 610
 - \@l@dttempcnta 433,
435, 437, 438, 442, 444, 446,
447, 1069, 1108, 1109, 1111,
1113, 1116, 1117, 1134–1138,
1140, 1147, 1152, 1156, 1164,
1169, 1173, 1206, 1209, 1211, 1215
 - \@l@dttempcntb . 172, 174, 176, 1099,
1100, 1147, 1152, 1156, 1164,
1169, 1173, 1198, 1202, 1215,
1223–1225, 1227, 1248–1250,
1252, 1269–1271, 1273, 1405,
1407, 1409, 1480–1482, 1484,
1636–1640, 1644–1647, 1651–1654
 - \@l@reg 331
 - \@l@regR 282
 - \@lab 552, 1354, 1366, 1391
 - \@lock 962, 1049
 - \@lockR . 64, 304, 306, 308, 321, 467,
483, 484, 486, 487, 515, 516,
518, 1000, 1031, 1075, 1077,
1078, 1080, 1161, 1178, 1180, 1182
 - \@lopL 576, 2027
 - \@lopR 576, 2030
 - \@nobreakfalse 824, 865
 - \@nobreaktrue 822, 826, 863, 867
 - \@nobptrue 1800, 1801, 1806, 1807
 - \@oldnobreak 822, 824, 863, 865, 912, 930
 - \@pbtrue 1798, 1799, 1804, 1805
 - \@pend 567, 1844
 - \@pendR 567, 1849
 - \@pstartsfalse 1817
 - \@pstartstrue 1817
 - \@ref 539, 614, 618
 - \@ref@reg 565
 - \@schapter 765
 - \@set 392, 648, 649, 1770, 1774
 - \@tag 678, 701, 1494
 - \@temp 1673
 - \@templ@d 1471, 1472
 - \@writelinesinparL .. 1766, 1842, 2148
 - \@writelinesinparR .. 1767, 1842, 2183
 - \@writelinesonpageL . 1922, 1924, 2026
 - \@writelinesonpageR . 1946, 1948, 2026
 - \@xloop 1330
- A**
- \absline@num
356, 426, 440, 459, 1040, 1796,
1804, 1806, 2081, 2082, 2085, 2106
 - \absline@numR 59, 233, 284,
287, 290, 423, 431, 452, 471,
505, 533, 544, 1022, 1061, 1062,
1099, 1316, 1797, 1805, 1807,
2093, 2094, 2097, 2117, 2214, 2216
 - \actionlines@list
..... 274, 277, 426, 440, 459
 - \actionlines@listR
..... 237, 252, 266, 269, 423,
431, 452, 471, 505, 533, 1121, 1124
 - \actions@list . 278, 427, 447, 461, 463
 - \actions@listR
. 237, 253, 270, 424, 438, 454,
456, 473, 482, 507, 514, 534, 1125
 - \add@inserts 968
 - \add@inserts@nextR 1305
 - \add@insertsR 1006, 1305
 - \add@penaltiesL 974, 1326
 - \add@penaltiesR 1012, 1326
 - \addtocontents 1683–1685
 - \addtocounter
. 914, 932, 1777, 1778, 2160, 2195
 - \advancelabel@refs 1361, 1373
 - \advanceline 640, 671
 - \affixline@num 966
 - \affixline@numR 1004, 1131
 - \affixpstart@numL 965, 1238
 - \affixpstart@numR 1003, 1238

- \affixside@note 969
 - \affixside@noteR 1007, [1458](#)
 - \appto 1464, 1465
 - \apptocmd 1713
 - \araw@textfalse [1829](#)
 - \araw@texttrue [1829](#)
 - astanza (environment) 8, [1568](#)
 - \AtBeginDocument ... 1387, 1658, 1693
- B**
- \ballast@count 1059, 1064
 - \bbl@main@language 1709, 1710
 - \bbl@set@language 1700, 1701
 - \beginnumbering .. 6, [37](#), 771, 789, 829
 - \beginnumberingR ... [50](#), 121, 789, 870
 - \bfseries 815, 818
 - \bypage@Rfalse [141](#), 156, 161
 - \bypage@Rtrue [141](#), 151
 - \bypstart@Rfalse [141](#), 152, 162
 - \bypstart@Rtrue [141](#), 157
- C**
- \c@ballast 1064
 - \c@chapter 104
 - \c@chapterR 104
 - \c@firstlinenumR [181](#), 1204
 - \c@firstsublinenumR [185](#), 1199
 - \c@linenumincrementR [181](#), 1204
 - \c@page 608, 610, 1978, 1984, 1987, 1994
 - \c@pstart 1376
 - \c@pstartL 815
 - \c@pstartR 818, 1364
 - \c@section 105
 - \c@sectionR 105
 - \c@sublinenumincrementR .. [185](#), 1199
 - \c@subsection 106
 - \c@subsectionR 106
 - \c@subsubsection 107
 - \c@subsubsectionR 107
 - \ch@ck@l@ckR [1131](#)
 - \ch@cksub@l@ckR [1131](#)
 - \ch@cksub@lockR 1200
 - \chapter 751, 752, 760
 - \chapterinpages [744](#), 752, 762
 - \chardef 1565
 - \check@goal 2053, 2067, [2124](#)
 - \check@pstarts
 - 1738, 1768, [1817](#), 1877, 1885, 1893
 - \checkpageL 1904, 1917, [2046](#)
 - \checkpageR 1929, 1941, [2046](#)
 - \checkpb@columns 1764, [1791](#)
 - \checkpbL 1919, [2079](#)
 - \checkpbR 1943, [2079](#)
 - \checkraw@text
 - 1744, 1761, [1829](#), 1901, 1953
 - \checkverseL 1762, 1918, [2102](#)
 - \checkverseR 1763, 1942, [2102](#)
 - \cleardoublepage 1990
 - \clearl@dleftpage 1928, [1976](#)
 - \clearl@drightpage 1952, [1976](#)
 - \cleartoevenpage [1976](#)
 - \cleartol@devenpage 1866, [1976](#)
 - \closeout 589, 595, 598, 602
 - \columnrulewidth 4, [1812](#)
 - \Columns 3, [1724](#)
 - \columnseparator 4, 1758, [1812](#)
 - \content 1417, 1431, 1445
 - \correcthangingL 971, [1546](#)
 - \correcthangingR 1009, [1546](#)
 - \countLline [940](#), 951
 - \countRline [940](#), 989
 - \critext 676
 - \csdimgdef 2306, 2307
 - \csexpandonce 1420, 1424, 1434, 1438,
 - 1448, 1452, 1496, 1500, 1516, 1527
 - \csuse 1515, 1526,
 - 2223, 2224, 2231, 2232, 2246–
 - 2248, 2253–2255, 2281, 2293, 2306
- D**
- \DeclareOption 8–10
 - \def@tempb 159
 - \dimdef 2262, 2268, 2269,
 - 2281, 2295, 2296, 2298, 2302, 2303
 - \dimen 627, 628, 632–634, 638
 - \dingdef 2264, 2265
 - \divide 1136
 - \do@actions 1041
 - \do@actions@fixedcodeR [1068](#)
 - \do@actions@nextR [1068](#)
 - \do@actionsR 1023, [1068](#)
 - \do@ballast 1042
 - \do@ballastR 1024, [1059](#)
 - \do@insidelineLhook 971, [983](#)
 - \do@insidelineRhook 983, 1009
 - \do@lineL [950](#), 1748, 1752, 1908
 - \do@lineLhook 955, [983](#)
 - \do@lineR [988](#), 1749, 1754, 1932
 - \do@lineRhook [983](#), 993
 - \do@lockoff [502](#)

`\do@lockoffL` 526
`\do@lockoffR` 502
`\do@lockon` 467
`\do@lockonL` 499
`\do@lockonR` 467
`\dolistloop` 1469
`\dummy@ref` 548

E

`\edfont@info` 727, 730, 736, 739
`\edlabel` 1352
`\edtext` 699
`\eledmac@error` 22, 25, 28, 30
`\eledmac@warning` 1470
`\empty` .. 85, 88, 266, 274, 689, 712,
 725, 734, 838, 879, 1121, 1203,
 1211, 1307–1309, 1320, 1331,
 1355, 1367, 2001, 2007, 2014, 2020
`\end@lemmas` 689, 690, 712, 713
`\endashchar` 1342
`\endgraf` 908, 926, 1732, 1869
`\endline@num` 555, 561
`\endlock` 660, 1574, 1583, 1588
`\endnumbering` 6, 40, 78, 125, 790
`\endnumberingR` 53, 78, 110, 120, 133, 790
`\endpage@num` 554, 561
`\endstanzaextra` 1590
`\endsub` 627
`\endsubline@num` 556, 562
`\enlargethispage` 1809
environments:
 `astanza` 8, 1568
 `Leftside` 5, 769
 `pages` 4, 744
 `pairs` 3, 744
 `Rightside` 5, 787
`\extensionchars` . 48, 70, 116, 130, 138

F

`\f@x@l@cksR` 1131
`\first@linenum@out@Rfalse` .. 584, 590
`\first@linenum@out@Rtrue` 584
`\firstlinenum` 5, 190
`\firstlinenum*` 5, 190
`\firstsublinenum` 5, 190
`\firstsublinenum*` 5, 190
`\fix@page` 327, 334
`\flag@end`
 . 611, 685, 695, 696, 708, 718, 719

`\flag@start`
 . 611, 684, 685, 696, 707, 708, 719
`\flush@notes` 1780, 1962
`\flush@notesR` 1329, 1781, 1963
`\footnotelang@lua` 1510, 1521
`\footnotelang@poly` 1514, 1525
`\fullstop` . 229, 1339, 1341, 1343, 1345

G

`\get@linelistfile` 262
`\get@nextboxL` 1916, 2137
`\get@nextboxR` 1940, 2137
`\getline@numL` 961, 1039
`\getline@numR` 999, 1021
`\getlinesfrompagelistL`
 1889, 1955, 2013
`\getlinesfrompagelistR`
 1890, 1956, 2013
`\getlinesfromparlistL` ... 1881, 2000
`\getlinesfromparlistR` ... 1882, 2000
`\gl@p` 269, 270,
 277, 278, 690, 713, 729, 738,
 1124, 1125, 1313, 1317, 1332,
 1358, 1370, 2004, 2010, 2017, 2023
`\goalfraction` 4, 2124

H

`\hangingsymbol` 9, 1537, 1543
`\hb@xt@` .. 964, 971, 978, 1002, 1009,
 1016, 1756, 1911, 1914, 1935, 1938
`\hsize` 856,
 896, 1756, 1911, 1914, 1935, 1938

I

`\if@filesw` 1681
`\if@firstcolumn` 1217, 1242, 1263, 1474
`\if@ledgroup` 594
`\if@nobreak` 821, 862
`\if@nopb` 1794, 1809
`\if@pb` 1793, 1810
`\if@pstarts` 1739, 1817, 1878, 1894
`\if@RTL` 685, 696, 708, 719
`\ifaraw@text` ... 1746, 1829, 1903, 1954
`\ifautopar` 848, 889
`\ifbypage@` 350
`\ifbypage@R` 141, 340, 1103
`\ifbypstart@` 569, 1769, 2153
`\ifbypstart@R` ... 141, 573, 1773, 2188
`\ifdim` 628, 632, 634,
 638, 1911, 1935, 1980, 2054, 2068

- \ifdimgreater 2284, 2287
 - \ifdimless 2297
 - \iffirst@linenum@out@R 584, 588
 - \ifinserthangingsymbol
..... 1535, 1549, 2105
 - \ifinserthangingsymbolR
..... 1533, 1541, 1559, 2116
 - \ifinstanzaL 767, 767, 1536, 1548, 2103
 - \ifinstanzaR 767, 768, 1542, 1558, 2114
 - \ifl@d@dash 1342
 - \ifl@d@elin 1344, 1345
 - \ifl@d@esl 1345
 - \ifl@d@pnum 1339, 1343
 - \ifl@d@ssub 1341
 - \ifl@dpagefull 1921, 1945, 2046
 - \ifl@dpadding 12, 1547, 1557
 - \ifl@dpairing 12, 82
 - \ifl@dsamelang 1669, 1747
 - \ifl@dsamepage 1907, 1931, 2046
 - \ifl@dskipnumber 1194
 - \ifl@dusedbabel 1667
 - \iflabelpstart 858, 898
 - \ifledgroupnotesL@ 1043
 - \ifledgroupnotesR@ 1025, 1193
 - \iflednopbinverse 2104, 2115
 - \ifledplinenum 1340
 - \ifledRcol 12, 173, 195,
199, 203, 207, 249, 264, 328,
337, 362, 376, 393, 410, 422,
430, 451, 496, 523, 532, 541,
612, 622, 629, 635, 641, 648,
656, 661, 665, 670, 680, 703,
724, 1353, 1392, 1406, 1419,
1433, 1447, 1495, 1508, 1704, 1714
 - \ifluatex 1509, 1520
 - \ifnoteschanged@ 92
 - \ifnumberedpar@ ... 831, 872, 904,
922, 1418, 1432, 1446, 1493, 1507
 - \ifnumbering 38, 146, 827, 901
 - \ifnumberingR ... 51, 79, 112, 868, 919
 - \ifnumberline 1025, 1043, 1193
 - \ifnumberpstart ... 849, 890, 913, 931
 - \ifnumequal 1462
 - \ifnumgreater 1470
 - \ifodd 1227, 1252,
1273, 1484, 1978, 1984, 1987, 1994
 - \ifparledgroup 2273, 2305
 - \ifpst@rtedL 33, 835
 - \ifpst@rtedR 33, 876
 - \ifpstartnum 1283, 1288
 - \ifpstartnumR 1238
 - \ifshiftedpstarts 5, 1910, 1934, 2125
 - \ifsidepstartnum 850, 891, 1240, 1261
 - \IfStrEq 959, 997, 1795,
1803, 2080, 2084, 2092, 2096,
2107, 2108, 2118, 2119, 2223,
2224, 2231, 2232, 2244, 2245, 2252
 - \ifstrequal 2276, 2282
 - \ifsublines@ 227,
316, 361, 394, 401, 432, 441,
453, 460, 472, 506, 560, 562,
1026, 1044, 1110, 1197, 1394, 1398
 - \ifvbox 952, 990, 1833, 1836, 2138, 2173
 - \ifvmode 1360, 1372
 - \ifwrittenlinesL 2134, 2146
 - \ifwrittenlinesR 2135, 2181
 - \initnumbering@reg 46
 - \initnumbering@sectcmd 74
 - \initnumbering@sectcountR 75, 99
 - \insert@count 538, 618, 681,
704, 1426, 1440, 1454, 1502, 1529
 - \insert@countR 539, 614, 680,
703, 1422, 1436, 1450, 1498, 1518
 - \insert@noterule@ledgroup
..... 2227, 2235, 2243
 - \inserthangingsymbolfalse 962
 - \inserthangingsymbolL ... 971, 1533
 - \inserthangingsymbolR ... 1009, 1533
 - \inserthangingsymbolRfalse 1000
 - \inserthangingsymbolRtrue 1000
 - \inserthangingsymboltrue 962
 - \insertlines@listR
.... 85, 237, 251, 544, 1309, 1313
 - \inserts@list
837, 1425, 1439, 1453, 1501, 1528
 - \inserts@listR
.. 878, 1304, 1307, 1317, 1331,
1332, 1421, 1435, 1449, 1497, 1517
 - \instanzaLfalse 1788, 1969
 - \instanzaLtrue 778
 - \instanzaRfalse 1789, 1970
 - \instanzaRtrue 802
 - \interlinepenalty 1576
 - \itemcount@ 1460, 1462, 1467, 1470
- L**
- \l@d@nums 727, 730, 736, 739
 - \l@d@set 409, 656, 657
 - \l@dbbl@set@language 1678, 1701
 - \l@dbfnote 1492

- \l@dc@maxchunks 843, 845,
884, 886, 1626, 1636, 1644, 1651
- \l@dcalc@maxoftwo
..... 1883, 2033, 2162, 2197
- \l@dcalc@minoftwo .. 1891, 1957, 2033
- \l@dcalcnum 1131
- \l@dchecklang 1671, 1745
- \l@dchset@num 283, 286, 409
- \l@dcsnote 1415
- \l@dcsnotetext 1469, 1472
- \l@demptyd@ta 956, 994
- \l@dend@stuff ... 49, 71, 117, 131, 139
- \l@dgetline@margin 171
- \l@dgetsidenote@margin 1404
- \l@dld@ta 967, 1005,
1218, 1230, 1243, 1255, 1264, 1276
- \l@dleftbox
.. 937, 963, 978, 1757, 1911, 1914
- \l@dlinenumR 219
- \l@dlsn@te 970, 1008
- \l@dlsnote 1415
- \l@dmake@labels 1376
- \l@dmake@labelsR 1364, 1381
- \l@dminpagelines
.... 1853, 1892, 1958, 2055, 2069
- \l@dnumpstartsL . 42, 842, 843, 845,
847, 1630, 1662, 1727, 1728,
1785, 1820, 1863, 1864, 1967, 2151
- \l@dnumpstartsR . 55, 883, 884, 886,
888, 1630, 1663, 1727, 1728,
1786, 1823, 1863, 1864, 1968, 2186
- \l@doldbbl@set@language 1700
- \l@doldselectlanguage 1699, 1703, 1708
- \l@dpagetrue
.... 2046, 2082, 2087, 2094, 2099
- \l@dpagetrue
.... 2046, 2081, 2086, 2093, 2098
- \l@dpagingfalse 14, 746, 759
- \l@dpagingtrue 754
- \l@dpairingfalse 12, 748, 758
- \l@dpairingtrue 745, 753
- \l@dpscL 952, 957, 1632,
1664, 1736, 1742, 1783, 1820,
1833, 1873, 1879, 1884, 1887,
1895, 1965, 2138, 2140, 2143,
2151, 2162, 2164, 2166, 2222, 2277
- \l@dpscR 990, 995,
1633, 1665, 1737, 1743, 1784,
1823, 1836, 1874, 1880, 1888,
1896, 1966, 2173, 2175, 2178,
2186, 2197, 2199, 2201, 2230, 2279
- \l@drd@ta 971, 1009,
1220, 1228, 1245, 1253, 1266, 1274
- \l@drightbox
..... 937, 1001, 1016, 1759, 1935, 1938
- \l@drsn@te 972, 1010
- \l@drsnote 1415
- \l@dsamelangfalse 1669, 1672
- \l@dsamelangtrue 1669, 1675
- \l@dsamepagetrue
.... 2046, 2081, 2086, 2093, 2098
- \l@dsamepagetrue
.... 2046, 2082, 2087, 2094, 2099
- \l@dsetupmaxlinecounts .. 1643, 1660
- \l@dsetupprawboxes 1635, 1659
- \l@dskipnumberfalse 1195
- \l@dskipnumbertrue 1091
- \l@dunhbox@line 971, 1009
- \l@dusedbabelfalse 1667, 1696
- \l@dusedbabeltrue 1667, 1698
- \l@duselanguage
.... 1688, 1751, 1753, 1905, 1930
- \l@dzeromaxlinecounts ... 1643, 1661
- \l@dzeropenalties 907, 925, 1731, 1868
- \l@prev@nopbR 62, 2209
- \l@prev@pbR 61, 2208
- \l@pscL 1632
- \l@pscR 1632
- \label@refs
1356, 1358, 1364, 1368, 1370, 1376
- \labelref@list 1367, 1370, 1399
- \labelref@listR 1350, 1355, 1358, 1395
- \language name .. 1679, 1680, 1682–1685
- \last@page@num 348, 354
- \last@page@numR 334
- \lastbox 960, 998
- \lastskip 627, 633
- \lcolwidth .. 4, 16, 755, 856, 964, 978
- \led@err@BadLeftRightPstarts ...
..... 24, 1728, 1864
- \led@err@LeftOnRightPage ... 27, 1988
- \led@err@LineationInNumbered ... 147
- \led@err@NumberingNotStarted ... 96
- \led@err@NumberingShouldHaveStarted
..... 119
- \led@err@NumberingStarted 39, 52
- \led@err@PendNoPstart 905, 923
- \led@err@PendNotNumbered ... 902, 920
- \led@err@PstartInPstart ... 832, 873

- \led@err@PstartNotNumbered . 828, 869
 - \led@err@RightOnLeftPage ... 27, 1995
 - \led@err@TooManyPstarts 21, 844, 885
 - \led@mess@NotesChanged 93
 - \led@mess@SectionContinued
 - 115, 129, 137
 - \led@nopbnumR 2213, 2214
 - \led@nopbR 2212, 2214
 - \led@pb@setting
 - 1795, 1803, 2080, 2084,
 - 2092, 2096, 2107, 2108, 2118, 2119
 - \led@pbnumR 2211, 2214
 - \led@pbR 2210, 2214
 - \led@warn@BadAction 1093
 - \led@warn@BadAdvancelineLine 379, 385
 - \led@warn@BadAdvancelineSubline .
 - 365, 371
 - \led@warn@BadLineation 164
 - \led@warn@BadSetline 646
 - \led@warn@BadSetlinenum 654
 - \led@warn@DuplicateLabel 1383
 - \ledgroupnotesL@false 2238
 - \ledgroupnotesL@true 2226
 - \ledgroupnotesR@false 2241
 - \ledgroupnotesR@true 2234
 - \ledllfill 971, 1009
 - \lednopb 798
 - \lednopbnum 2107, 2210
 - \lednopbnumR 2118, 2210
 - \lednopbR 798, 2212
 - \ledpb 797
 - \ledpbnum 2108
 - \ledpbnumR 2119, 2210
 - \ledpbR 797, 2210
 - \ledRcolfalse 15, 770, 804
 - \ledRcoltrue 788
 - \ledrlfill 971, 1009
 - \ledsavedprintlines 7, 1337
 - \ledstrutL 1911, 1914, 1973
 - \ledstrutR 1935, 1938, 1973
 - \ledthegoal 2054, 2068, 2124
 - \leftlinenumR 219, 1218, 1230
 - \leftpstartnumL 1238
 - \leftpstartnumR 1238
 - Leftside (environment) 5, 769
 - \Leftsidehook 776, 782
 - \Leftsidehookend 781, 782
 - \line@list 734, 738
 - \line@list@stuff 48, 130
 - \line@list@stuffR .. 70, 116, 138, 586
 - \line@listR . 88, 237, 250, 562, 725, 729
 - \line@margin 176, 1248
 - \line@marginR 169, 1223, 1269
 - \line@num . 351, 383, 384, 386, 404,
 - 415, 416, 444, 569, 1050, 1397, 2156
 - \line@numR 63, 226, 233,
 - 288, 322, 341, 377, 378, 380,
 - 397, 411, 412, 435, 555, 559,
 - 573, 1032, 1104, 1113, 1202,
 - 1204, 1206, 1207, 1393, 1470, 2191
 - \lineation 799
 - \lineationR 145, 799
 - \linenum@out 608, 617,
 - 625, 630, 636, 642, 649, 657,
 - 662, 666, 1366, 1770, 1844, 2027
 - \linenum@outR 583, 589,
 - 591, 595, 596, 598, 599, 602,
 - 603, 610, 613, 623, 629, 635,
 - 641, 648, 656, 661, 665, 670,
 - 1354, 1774, 1849, 2030, 2210–2213
 - \linenumberlist 1203, 1207
 - \linenumincrement 5, 190
 - \linenumincrement* 5, 190
 - \linenummargin 169
 - \linenumr@p 1340, 1344, 1393, 1397
 - \linenumrepR 216, 226
 - \linenumsep
 - . 221, 223, 1285, 1288, 1297, 1300
 - \linesinpar@listL
 - 242, 258, 570, 2001, 2004
 - \linesinpar@listR
 - 242, 254, 574, 2007, 2010
 - \linesonpage@listL 259, 578, 2014, 2017
 - \linesonpage@listR 255, 581, 2020, 2023
 - \list@clear
 - . 250–255, 258, 259, 261, 837, 878
 - \list@clearing@reg 257
 - \list@create
 - ... 237–240, 242–244, 1304, 1350
 - \listcsxadd 356, 2214–2217
 - \lock@disp 1163, 1167, 1172
 - \lock@off 493, 494, 502, 665, 666
 - \lock@on 661, 662
- M
- \managestanza@modulo 1602
 - \maxchunks 3, 1626
 - \maxlinesinpar@list 242, 261
 - \memorydump 6, 775, 793
 - \memorydumpL 124, 775

\memorydumpR 124, 793
 \message 47, 69
 \multiply 1137

N

\n@num 530, 670
 \n@num@reg 536
 \namebox 952, 957, 990,
 995, 1610, 1833, 1836, 2138, 2173
 \NeedsTeXFormat 2
 \new@lineL 607, 971
 \new@lineR 609, 1009
 \newbox 810, 937, 938, 1611
 \newcounter 99–102, 181,
 183, 185, 187, 813, 814, 816, 817
 \newif 5, 13, 34,
 141, 142, 584, 767, 768, 1292,
 1533, 1667, 1669, 1793, 1794,
 1817, 1829, 2046, 2048, 2134, 2135
 \newmarks 2218–2220
 \newnamebox 1610, 1638, 1639
 \newnamecount 1621, 1646
 \newwrite 583
 \next@absline
 1796, 1798, 1800, 2085–2087
 \next@abslineR
 1797, 1799, 1801, 2097–2099
 \next@action 278
 \next@actionline 275, 277
 \next@actionlineR
 . 267, 269, 1062, 1100, 1122, 1124
 \next@actionR 270, 1063,
 1101, 1102, 1107, 1108, 1116, 1125
 \next@insert 838
 \next@insertR
 879, 1308, 1311, 1313, 1316, 1320
 \next@page@num 355, 427
 \next@page@numR 67, 291, 293, 345, 424
 \no@expands 678, 701
 \normal@page@breakR 60
 \normal@pars 81, 841, 882
 \normalbfnoteX 1506
 \notefontsetup 2264
 \noteschanged@true
 86, 89, 726, 735, 1310
 \num@lines 908, 1732, 1869
 \num@linesR 809, 926, 1733, 1870
 \numberedpar@true 857, 897
 \numberingRfalse 80
 \numberingRtrue 57, 110, 134

\numberingtrue 44, 126
 \numberpstartfalse 7
 \numberpstarttrue 7
 \numdef 1460, 1467,
 1796, 1797, 2085, 2097, 2277, 2279
 \numgdef 2106, 2117
 \numlabfont 226
 \numpagelinesL 1853,
 1909, 1922, 1926, 2055, 2108, 2297
 \numpagelinesR
 1853, 1933, 1946, 1950, 2069, 2119

O

\oldchapter 751, 760
 \oldstanza 777, 778, 780, 801, 802, 805
 \one@line 957, 960, 971
 \one@lineR 809, 995, 998, 1009
 \openout 591, 596, 599, 603

P

\p@pstartL 859
 \p@pstartR 899
 \page@action 292, 421, 549
 \page@num 273, 353, 1250, 1482
 \page@numR 246, 265, 343,
 554, 559, 1102, 1225, 1271, 1470
 \pagebreak 1810
 \pagegoal 2132
 \Pages 4, 1857
 pages (environment) 4, 744
 \pagetotal 1980, 2054, 2068
 pairs (environment) 3, 744
 \par@line 909, 1734, 1871
 \par@lineR 809, 927, 1735, 1872
 \parledgroup@ 959, 997, 2218, 2244
 \parledgroup@beforenotes@save ..
 916, 934, 2304
 \parledgroup@beforenotesL 2302
 \parledgroup@beforenotesR 2302
 \parledgroup@correction@notespacing
 1913, 1937, 2291
 \parledgroup@correction@notespacing@final
 2168, 2203, 2272
 \parledgroup@correction@notespacing@init
 1927, 1951, 2267, 2275
 \parledgroup@notes@endL
 2141, 2144, 2167, 2237
 \parledgroup@notes@endR
 2176, 2179, 2202, 2240

- `\parledgroup@notes@startL`
 959, [2221](#), [2237](#)
`\parledgroup@notes@startR`
 997, [2221](#), [2237](#)
`\parledgroup@notespacing@correction`
 [2262](#), 2294–2296
`\parledgroup@notespacing@correction@accumulated`
 2268, 2274, 2295
`\parledgroup@notespacing@correction@modulight`
 2269, 2296–2298
`\parledgroup@notespacing@set@correction`
 1860, [2262](#)
`\parledgroup@series`
 2219, 2223, 2224,
 2231, 2232, 2247, 2248, 2254, 2255
`\parledgroup@type` 2220,
 2223, 2224, 2231, 2232, 2245, 2252
`\parledgroupnotespacing` . [2261](#), 2264
`\parledgroupseries@` [2218](#)
`\parledgrouptrue` 10
`\parledgrouptype@` [2218](#)
`\pausenumbering` 791
`\pausenumberingR` [109](#), 791
`\pend` 5, 774, 796, 833, 1589
`\pendL` 774, [901](#)
`\pendR` 796, 874, [919](#)
`\prev@abslineverse`
 2106–2108, 2117–2119
`\prev@nopbR` [2208](#)
`\prev@pbR` [2208](#)
`\prevgraf`
 . 908, 926, 1732, 1733, 1869, 1870
`\printlines` 1348
`\printlinesR` 7, [1337](#)
`\ProcessOptions` 11
`\protected@csxdef` 1515, 1526
`\protected@edef` 858, 898
`\protected@write` ... 1363, 1375, 1682
`\ProvidesPackage` 3
`\pst@rtedLfalse` 33, 43
`\pst@rtedLtrue` 127, 839
`\pst@rtedRfalse` 35, 56, 83
`\pst@rtedRtrue` 113, 135, 880
`\pstart` 5, 22, 26, 772, 795, 1591
`\pstartL` 772, [812](#)
`\pstartnumfalse` 1285, 1290
`\pstartnumRfalse` 1297, 1302
`\pstartnumRtrue` 1293, 1741, 2196
`\pstartnumtrue` 1740, 2161
`\pstartR` 795, [812](#)
- R**
- `\Rcolwidth` . 4, [16](#), 756, 896, 1002, 1016
`\read@linelist` [248](#), 587
`\rem@inder` 1207, 1209–1211
`\resetprevline@` 1771, 1775, 2157, 2192
`\resumenumbering` 792
`\resetnumberingR` [109](#), 792
`\rightlinenumR` [219](#), 1220, 1228
`\rightpstartnumL` [1238](#)
`\rightpstartnumR` [1238](#)
`\Rightside (environment)` 5, [787](#)
`\Rightsidehook` [782](#), 800
`\Rightsidehookend` [782](#), 806
`\rlap` 1220, 1228, 1245, 1253, 1266, 1274
`\Rlineflag` 7, [214](#), 226, 1340, 1344, 1385
`\rule` 1813
- S**
- `\sc@n@list` 1208, 1210
`\secdef` 765
`\section@num` 45, 47, 48, 128–130
`\section@numR`
 . [31](#), 58, 69, 70, 114–116, 136–138
`\select@language` ... 1680, 1682–1685
`\selectlanguage` [1688](#)
`\set@line` 679, 702, [723](#)
`\set@line@action`
 285, 390, 399, 406, [429](#), 551
`\set@dlp@rbox` 1475, 1487
`\set@drp@rbox` 1477, 1485
`\setline` [644](#)
`\setlinenum` [652](#)
`\setnamebox` 847, 888, [1610](#)
`\setprintlines` 1338
`\shiftedpstartsfalse` 7
`\shiftedpstartstrue` 6, 8, 9
`\shiftedversesfalse` 7
`\shiftedversestrue` 6
`\showlemma` 688, 711
`\sidenote@margin` 1409, 1413
`\sidenote@marginR` [1402](#), 1480
`\sidenotecontent@`
 1459, 1464, 1465, 1475, 1477, 1487
`\sidenotecontent@t` 1485
`\sidenotemargin` [1402](#)
`\sidenotessep` 1465
`\skip` 2247, 2254
`\skip@lockoff` 494, [502](#)
`\skipnumbering` 8, [669](#)
`\skipnumbering@reg` 673

- \smash 1813
 - \splitbotmarks 2244,
2245, 2247, 2248, 2252, 2254, 2255
 - \splitfirstmarks
. 959, 997, 2223, 2224, 2231, 2232
 - \splittopskip 954, 992
 - \stanza ... 777, 778, 780, 801, 802, 805
 - \stanza@count 1571, 1585, 1597
 - \stanza@hang 1573, 1605
 - \stanza@modulo 1571, 1600
 - \stanzaindentbase
..... 1551, 1561, 1598, 1601
 - \startlock 660
 - \startstanzahook 1569
 - \startsub 627
 - \sub@action 301, 450, 550
 - \sub@change 68, 295, 296, 302
 - \sub@lock 1045
 - \sub@lockR 65, 310, 312, 314,
317, 468, 474, 475, 477, 478,
508, 509, 511, 1027, 1083, 1085,
1086, 1088, 1144, 1184, 1186, 1188
 - \sub@off 635, 636
 - \sub@on 629, 630
 - \subline@num 228, 351, 369,
370, 372, 402, 442, 1046, 1051, 1398
 - \subline@numR 229,
233, 318, 322, 341, 363, 364,
366, 395, 433, 556, 560, 1028,
1033, 1104, 1111, 1198, 1199, 1394
 - \sublinenumincrement 5, 190
 - \sublinenumincrement* 5, 190
 - \sublinenumr@p . 1341, 1345, 1394, 1398
 - \sublinenumrepR 216, 229
 - \sublines@false 66, 299, 1073
 - \sublines@true 297, 1071
 - \sublock@disp 1146, 1150, 1155
 - \symplinenum 1340
 - \sza@penalty 1580, 1584
- T**
- \temp@spacing 2264, 2265
 - \textwidth 17, 19, 755, 756
 - \theledlanguageL 1673, 1688, 1751, 1905
 - \theledlanguageR 1673, 1688, 1753, 1930
 - \thepage 608, 610, 1364, 1376
 - \thepstart 773, 794
 - \thepstartL
. 7, 773, 815, 852, 859, 1284, 1289
 - \thepstartR
. 7, 794, 818, 892, 899, 1296, 1301
- \thr@@ .. 477, 486, 509, 516, 1078, 1086
 - \topskip 1980
- U**
- \unhbox 1615,
1757, 1759, 1911, 1914, 1935, 1938
 - \unhnamebox 1610
 - \unvbox 960, 998, 1617
 - \unvnamebox 1610
 - \usernamecount 1572, 1579, 1621, 1653,
1884, 2140, 2143, 2162, 2164,
2175, 2178, 2197, 2199, 2222, 2230
- V**
- \value 836, 877, 1596,
1725, 1726, 1858, 1859, 2154, 2189
 - \vbadness 953, 991
 - \vbfnoteX 1516, 1527
 - \vbox 847, 888
 - \vl@dbfnote 1496, 1500
 - \vl@dcnote 1448, 1452
 - \vl@dlnote 1420, 1424
 - \vl@drsnote 1434, 1438
 - \vsplit 957, 995
- W**
- \wd 971, 1009
 - \WithSuffix 210–213
 - \writtenlinesLfalsed 1875, 2152
 - \writtenlinesLtrue 2149
 - \writtenlinesRfalse 1876, 2187
 - \writtenlinesRtrue 2184
- X**
- \x@lemma 690–692, 713–715
 - \xifinlistcs 1798–
1801, 1804–1807, 2081, 2082,
2086, 2087, 2093, 2094, 2098, 2099
 - \xpg@main@language 1718, 1719
 - \xpg@set@language 1713, 1717
 - \xright@appenditem
..... 423, 424, 426, 427, 431,
438, 440, 447, 452, 454, 456,
459, 461, 463, 471, 473, 482,
505, 507, 514, 533, 534, 544,
558, 570, 574, 578, 581, 1393,
1397, 1420, 1424, 1434, 1438,
1448, 1452, 1496, 1500, 1516, 1527
- Z**
- \zz@@@ 1356, 1368

Change History

v0.1		
General: First public release	1	
v0.10		
General: <code>\edlabel</code> commands on the right side are now correctly indicated.	1	
<code>\edlabel</code> commands which start a paragraph are now put in the right place.	1	
v0.11		
General: Change <code>\do@lineL</code> and <code>\do@lineR</code> to allow line numbering by <code>pstart</code> (like in <code>eledmac</code> 0.15).	40	
Lineation can be by <code>pstart</code> (like in <code>eledmac</code> 0.15).	16	
New management of hangingsymbol insertion, preventing undesirable insertions.	55	
Prevent shift of column separator when a verse is hanged	55	
<code>\affixline@numR</code> : Changed <code>\affixline@numR</code> to allow to disable line numbering (like in <code>eledmac</code> 0.15).	44	
<code>\Columns</code> : Line numbering by <code>pstart</code>	62	
<code>\get@nextboxR</code> : Change <code>\get@nextboxL</code> and <code>\get@nextboxR</code> to allow to disable line numbering (like in <code>eledmac</code> 0.15).	72	
<code>Pstart</code> number can be printed in side	73	
v0.12		
General: New new management of hangingsymbol insertion, preventing undesirable insertions.	55	
v0.2		
General: Added section of babel related code	59	
Fix babel problems	1	
<code>\Columns</code> : Added <code>\l@dchecklang</code> and <code>\l@duselanguage</code> to		
<code>\Columns</code>	62	
<code>\Pages</code> : Added <code>\l@duselanguage</code> to <code>\Pages</code>	66	
v0.3		
General: Added <code>\do@lineLhook</code> and <code>\do@lineRhook</code>	40	
Reorganize for <code>ledarab</code>	1	
<code>\affixline@numR</code> : Changed <code>\affixline@numR</code> to match new <code>eledmac</code>	44	
<code>\do@actions@nextR</code> : Used <code>\do@actions@fixedcode</code> in <code>\do@actionsR</code>	42	
<code>\do@lineL</code> : Added <code>\do@lineLhook</code> to <code>\do@lineL</code>	40	
Simplified <code>\do@lineL</code> by using macros for some common code	40	
<code>\do@lineR</code> : Changed <code>\do@lineR</code> similarly to <code>\do@lineL</code>	41	
<code>Leftside</code> : Added hooks into <code>Leftside</code> environment	34	
<code>\flag@end</code> : Removed extraneous spaces from <code>\flag@end</code>	30	
<code>\ifledRcol</code> : Moved <code>\ifl@dpairing</code> to <code>eledmac</code>	13	
<code>\ifpst@rtedR</code> : Moved <code>\ifpst@rtedL</code> to <code>eledmac</code>	14	
<code>\l@ddlinenumR</code> : Simplified <code>\leftlinenumR</code> and <code>\rightlinenumR</code> by introducing <code>\l@ddlinenumR</code>	19	
<code>\l@dnumpstartsR</code> : Moved <code>\l@dnumpstartsL</code> to <code>eledmac</code>	58	
<code>\ledsavedprintlines</code> : Simplified <code>\printlinesR</code> by using <code>\setprintlines</code>	50	
<code>\ledstrutR</code> : Added <code>\ledstrutL</code> and <code>\ledstrutR</code>	68	
<code>\normalbfnoteX</code> : Removed extraneous spaces from <code>\normalbfnoteX</code>	54	
<code>\Pages</code> : Added <code>\ledstrutL</code> to <code>\Pages</code>	66	
Added <code>\ledstrutR</code> to <code>\Pages</code>	67	

	<code>\Rightsidehookend:</code> Added	Possibility to number the	
	<code>\Leftsidehook, \Leftsidehookend,</code>	<code>pstart</code> with the commands	
	<code>\Rightsidehook</code> and <code>\Rightsidehookend</code>	<code>\numberpstarttrue</code>	1
 35	<code>\ifledRcol:</code> Moved <code>\iflledRcol</code>	
	<code>\sublinenumrepR:</code> Added	and <code>\ifnumberingR</code> to <code>eledmac</code>	13
	<code>\linenumrepR</code> and <code>\sublinenumrepR</code>	v0.9.1	
 19	General: The numbering of the	
v0.3.a	General: Minor <code>\linenummargin</code>	<code>pstarts</code> restarts on each	
	fix 1	<code>\beginnumbering</code>	1
v0.3.b	General: Improved parallel page	v0.9.2	
	balancing 1	General: Debug : with <code>\Columns</code> ,	
v0.3.c	General: Compatibility with Poly-	the hanging indentation now	
	glossia 1	runs on the left columns and the	
v0.3.a	General: Compatibility with Poly-	hanging symbol is shown only	
	glossia 1	when <code>\stanza</code> is used.	1
v0.3a	<code>\line@marginR:</code> Don't just	v0.9.3	
	set <code>\line@marginR</code> in	General: <code>\thepstartL</code> and	
	<code>\linenummargin</code> 17	<code>\thepstartR</code> use now	
v0.3b	<code>\Pages:</code> Added <code>\l@dminpagelines</code>	<code>\bfseries</code> and not <code>\bf</code> , which	
	calculation for succeeding page	is deprecated and makes con-	
	pairs 67	licts with memoir class.	1
v0.4	General: No more <code>ledparpatch</code> . All	v1.0	
	patches are now in the main	General: Compatibility with <code>eled-</code>	
	file. 1	<code>mac</code> . Change name to <code>eledpar</code> . .	1
v0.5	General: Corrections about	Debug in lineation by <code>pstart</code> ..	16
	<code>\section</code> and other titles in	v1.0.1	
	numbered sections 1	General: Correction on <code>\numberonlyfirstinline</code>	
v0.6	General: Be able to use <code>\chapter</code> in	with lineation by <code>pstart</code> or by	
	parallel pages. 1	page. 1	
v0.7	General: Option 'shiftedverses'	v1.1	
	which make there is no blank	General: <code>Shiftedverses</code> becomes	
	inequal length. 1	<code>shiftedpstarts</code> 1	
v0.8	General: Possibility to have a sym-	<code>\pstartR:</code> Add <code>\labelpstarttrue</code>	
	bol on each hanging of verses,	(from <code>eledmac</code>). 36	
	like in the french typogra-	v1.1.1	
	phy. Redefine the commande	<code>\pstartR:</code> Correct <code>\pstartR</code> bug in-	
	<code>\hangingsymbol</code> to define the	troduced by 1.1. 36	
	character. 1	v1.1.2	
v0.9	General: Possibility to number	<code>\affixside@noteR:</code> Remove spuri-	
	<code>\pstart</code> 7	ous space between line number	
		and line content 53	
		v1.2	
		General: Support for <code>\led<section></code>	
		commands in parallel texts. . . .	1
		v1.2.1	
		<code>\initnumbering@sectcountR:</code> For	
		the right section, the counter is	
		defined only once. 15	
		v1.3	
		General: Manage RTL language. .	32

v1.3.2	General: Debug with some classes.	1	when in minipage, except if the minipage is a ledgroup.	29
v1.3.3	\l@dbfnote: Spurious space with footnote in right column.	54	v1.4.3	General: Corrects a false hanging verse when a verse is exactly the length of a line.
	\l@dcnote: Debug on the left notes of the right column.	52		1
v1.3.4	\l@dcnote: Allow to use commands in sidenotes, like it was introduced by eledmac 1.0.	52	\inserthangingsymbolR: Hang verse is now not automatically flush right.	55
v1.3.5	\normalbfnoteX: Allows one to redefine \thefootnoteX with alph when some packages are loaded.	54	\pendL: Spurious spaces in \pendL.	38
v1.4	General: Added \do@insidelineLhook and \do@insidelineRhook	40	\pendR: Spurious spaces in \pstartR.	39
v1.4.1	\normalbfnoteX: Fix bug with normal familiar footnotes when mixing RTL and LTR text.	54	\pstartR: Spurious spaces in \pstartL and \pstartR.	36
	astanza: Enable the use of stanzaindentrepetition within astanza environment.	56	v1.5.0	General: Add, as in eledmac, features to manage page breaks.
v1.4.2	\line@list@stuffR: Open / close immediatly the line-list file			1
			\sublinenumincrement*: Add starred version of \firstlinenum, \linenumincrement, \firstsublinenum, \sublinenumincrement to change both Left and Right-side.	18
			v1.6.0	General: Add tool and documentation for parallel ledgroups
				9
			v1.7.0	General: Add, as in eledmac, features to make crossrefs with pstart numbers.
				1