Parallel typesetting for critical editions: the eledpar package*

Peter Wilson
Herries Press†
Maïeul Rouquette‡

Abstract

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

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

eledpar provides many tools and options. Normally, they are all documented in this file. Also provided is a help folder, "examples". The folder contains additional examples (although not for all cases).

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 (better).

You can subscribe to the eledmac email list in:
`http://geekographie.maieul.net/146`

Contents

1 Introduction **3**

2 The eledpar package **4**
2.1 General 4

3 Parallel columns **5**

4 Facing pages **6**

*This file (`eledpar.dtx`) has version number v1.8.2, last revised 2014/08/15.

†`herries dot press at earthlink dot net`
‡`maieul at maieul dot net`

5 Left and right texts **7**

6 Numbering text lines and paragraphs **8**

7 Verse **10**

8 Side notes **12**

9 Parallel ledgroups **12**
9.1 Parallel ledgroups and setspace package 13

10 Sectioning commands **14**

11 Implementation overview **14**

12 Preliminaries **14**
12.1 Messages 15

13 Sectioning commands **16**

14 Line counting **18**
14.1 Choosing the system of lineation 18
14.2 Line-number counters and lists 22
14.3 Reading the line-list file 22
14.4 Commands within the line-list file 23
14.5 Writing to the line-list file 31

15 Marking text for notes **34**

16 Parallel environments **36**

17 Paragraph decomposition and reassembly **37**
17.1 Boxes, counters, \pstart and \pend 38
17.2 Processing one line 41
17.3 Line and page number computation 44
17.4 Line number printing 47
17.5 Pstart number printing in side 49
17.6 Add insertions to the vertical list 51
17.7 Penalties 51
17.8 Printing leftover notes 52

18 Footnotes **53**
18.1 Normal footnote formatting 53

19 Cross referencing **53**

20 Side notes **55**

21 Familiar footnotes **56**

22 Verse **57**

23 Naming macros **59**

24 Counts and boxes for parallel texts **60**

25 Fixing babel **61**

26 Parallel columns **63**

27 Parallel pages **68**

28 Sections' titles' commands **78**

29 Page break/no page break, depending on the specific line **78**

30 Parallel ledgroup **79**

31 The End **82**

Appendix A Some things to do when changing version **83**
Appendix A.1 Migration to eledpar 1.4.3 83

References **83**

Index **83**

Change History **93**

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 11 through 31); 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 11, 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 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 \Rcolwidth The lengths \Lcolwidth and \Rcolwidth are the widths of the left and right columns, respectively. By default, these are:
\setlength{\Lcolwidth}{0.45\textwidth}
\setlength{\Rcolwidth}{0.45\textwidth}
They may be adjusted if one text tends to be 'bulkier' than the other.

\columnrulewidth
\columnseparator

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

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

You can also modify \columnseparator if you want more control.

\columnsposition

By default, columns are positioned to the right of the page. However, you use \columnsposition{L} to align them to the left, or \columnsposition{C} to center them.

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.

\beforecolumnseparator
\aftercolumnseparator

By default, the spaces around column separator are the same as the space:

- On the left of columns, if columns are aligned right.
- On the right of columns, if columns are aligned left.
- On both the Left and Right columns, if columns are centered.

You can redefine \beforecolumnseparator and \aftercolumnseparator length to define spaces before or after the column separator, instead of letting eledpar calculate them automatically.

```
\setlength{\beforecolumseparator}{length}
\setlength{\aftercolumseparator}{length}
```

If you want to come back to the previous behavior, just set them with a negative value.

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

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

`\pstart` `\pend` 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

[1] when used with ledpatch v0.2 or greater.

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. `\lineationR` macro is the equivalent of eledmac `\lineation` macro for the right side. `\lineation*` macro is the equivalent of eledmac `\lineation` macro for both sides. 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.

`\lineationR`
`\lineation*`

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

Each section of numbered text must be preceded by `\beginnumbering` and followed by `\endnumbering`, like:

`\beginnumbering`
⟨*text*⟩

`\beginnumbering`
`\endnumbering`

`\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 `\endumbering` 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:
`\printlinesR` `\renewcommand*{\Rlineflag}{}`. The `\printlines` macro is ordinarily used
`\ledsavedprintlines` 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
\numberpstartfalse
\thepstartL
\thepstartR

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 `\sza@0@`' it is because you have forgotten to use `\setstanzaindents` to set the stanza indents.

\skipnumbering

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

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

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

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

```
\setstanzaindents{1,0,1,0,1,0,1,0,1,0,1}
```

```
\begin{pairs}
\begin{Leftside}
  \firstlinenum{2}
  \linenumincrement{1}
  \beginnumbering
  \begin{astanza}
    \stanzanum{1} First in first stanza &
                  Second in first stanza &
                  Second in first stanza &
                  Third in first stanza &
                  Fourth in first stanza &
    \interstanza
    \setline{2}\stanzanum{2} First in second stanza &
                  Second in second stanza &
                  Second in second stanza &
                  Third in second stanza &
                  Fourth in second stanza \&
  \end{astanza}
  ...
```

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

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

`\hangingsymbol` Like in eledmac, you could redefine the command `\hangingsymbol` to insert a character in each hanged line. If you use it, you must run LaTeXtwo 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 Side notes

As in eledmac, you must use one of the following commands to add side notes: `\ledsidenote`, `\ledleftnote`, `\ledrightnote`, `\ledouterote`, `\ledinnerrote`.

The `\sidenotemargin` defines the margin of the sidenote for either left or right side, depending on the current environment. You can use `\sidenotemargin*` to define it for both sides.

9 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 `\parledgrouptrue`.
- 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}
```

9.1 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.

10 Sectioning commands

The standard sectioning commands of eledmac are available, and provide parallel sectionings, for both two-column and two-page layout. By default, the section commands of the right side are not added to the table of contents. But you can change it, using `\eledsectnotoc{`⟨*arg*⟩`}`, where ⟨*arg*⟩ could be `L` (for left side) or `R` (for right side).

`\eledsectnotoc`

`\eledsectmark`

By default, the LaTeX marks for header are token from left side. You can change it, using `\eledsectmark{`⟨*arg*⟩`}`, where ⟨*arg*⟩ could be `L` (for left side) or `R` (for right side).

11 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.

12 Preliminaries

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

```
⟨*code⟩
\NeedsTeXFormat{LaTeX2e}
\ProvidesPackage{eledpar}[2014/08/15 v1.8.2 eledmac extension for parallel texts]%

```

With the option 'shiftedpstarts' a long pstart one 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

```
\newif\ifshiftedpstarts
\let\shiftedversestrue\shiftedpstartstrue
\let\shiftedversesfalse\shiftedpstartsfalse
\DeclareOption{shiftedverses}{\shiftedpstartstrue}
\DeclareOption{shiftedpstarts}{\shiftedpstartstrue}
\DeclareOption{parledgroup}{\parledgrouptrue}
\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@dpaging
\ifledRcol

`\ifl@dpairing` is set TRUE if we are processing parallel texts and `\ifl@dpaging` is also set TRUE if we are doing parallel pages. `\ifledRcol` is set TRUE if we are doing the right hand text. `\ifl@dpairing` is defined in eledmac.

```
  \l@dpairingfalse
\newif\ifl@dpaging
  \l@dpagingfalse
  \ledRcolfalse
```

\Lcolwidth
\Rcolwidth

The widths of the left and right parallel columns (or pages).

```
\newdimen\Lcolwidth
  \Lcolwidth=0.45\textwidth
\newdimen\Rcolwidth
  \Rcolwidth=0.45\textwidth

```

12.1 Messages

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

\eledpar@error

```
\newcommand{\eledpar@error}[2]{\PackageError{eledpar}{#1}{#2}}
%     \end{macrocode}
% \end{macro}
% \begin{macro}{\led@err@TooManyPstarts}
%    \begin{macrocode}
\newcommand*{\led@err@TooManyPstarts}{%
  \eledpar@error{Too many \string\pstart\space without printing.
                 Some text will be lost}{\@ehc}}
```

d@err@BadLeftRightPstarts

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

\led@err@LeftOnRightPage
\led@err@RightOnLeftPage

```
\newcommand*{\led@err@LeftOnRightPage}{%
  \eledpar@error{The left page has ended on a right page}{\@ehc}}
\newcommand*{\led@err@RightOnLeftPage}{%
  \eledpar@error{The right page has ended on a left page}{\@ehc}}
```

13 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.

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

\ifpst@rtedL
\ifpst@rtedR

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

```
  \pst@rtedLfalse
\newif\ifpst@rtedR
  \pst@rtedRfalse

```

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

```
\newcommand*{\beginnumberingR}{%
  \ifnumberingR
    \led@err@NumberingStarted
    \endnumberingR
  \fi
  \global\l@dnumpstartsR \z@
  \global\pst@rtedRfalse
  \global\numberingRtrue
  \global\advance\section@numR \@ne
  \global\absline@numR \z@
  \gdef\normal@page@breakR{}
  \gdef\l@prev@pbR{}
  \gdef\l@prev@nopbR{}
  \global\line@numR \z@
  \global\@lockR \z@
  \global\sub@lockR \z@
  \global\sublines@false
  \global\let\next@page@numR\relax
  \global\let\sub@change\relax
  \message{Section \the\section@numR R }%
  \line@list@stuffR{\jobname.\extensionchars\the\section@numR R}%
```

```
  \l@dend@stuff
  \setcounter{pstartR}{1}
  \begingroup
  \initnumbering@sectcountR
  \gdef\eled@sectionsR@@{}
  \makeatletter\InputIfFileExists{\jobname.eledsec\the\section@numR R}{}{}\makeatother
  \immediate\openout\eled@sectioningR@out=\jobname.eledsec\the\section@numR R\relax
}
```

`\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.

```
\def\endnumberingR{%
  \ifnumberingR
    \global\numberingRfalse
    \normal@pars
    \ifl@dpairing
      \global\pst@rtedRfalse
    \else
      \ifx\insertlines@listR\empty\else
        \global\noteschanged@true
      \fi
      \ifx\line@listR\empty\else
        \global\noteschanged@true
      \fi
    \fi
    \ifnoteschanged@
      \led@mess@NotesChanged
    \fi
  \else
    \led@err@NumberingNotStarted
  \fi
  \endgroup
  \immediate\closeout\eled@sectioningR@out
  }

```

`\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`.

```
\newcounter{chapterR}
\newcounter{sectionR}
\newcounter{subsectionR}
\newcounter{subsubsectionR}
\newcommand{\initnumbering@sectcountR}{
    \let\c@chapter\c@chapterR
    \let\c@section\c@sectionR
```

```
    \let\c@subsection\c@subsectionR
    \let\c@subsubsection\c@subsubsectionR
}
```

\pausenumberingR
\resumenumberingR

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

```
\newcommand*{\pausenumberingR}{%
  \endnumberingR\global\numberingRtrue}
\newcommand*{\resumenumberingR}{%
  \ifnumberingR
     \global\pst@rtedRtrue
     \global\advance\section@numR \@ne
     \led@mess@SectionContinued{\the\section@numR R}%
     \line@list@stuffR{\jobname.\extensionchars\the\section@numR R}%
     \l@dend@stuff
     \initnumbering@sectcmd
  \else
    \led@err@numberingShouldHaveStarted
     \endnumberingR
     \beginnumberingR
  \fi}

```

\memorydumpL
\memorydumpR

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

```
\newcommand*{\memorydumpL}{%
  \endnumbering
  \numberingtrue
  \global\pst@rtedLtrue
  \global\advance\section@num \@ne
     \led@mess@SectionContinued{\the\section@num}%
  \line@list@stuff{\jobname.\extensionchars\the\section@num}%
  \l@dend@stuff}
\newcommand*{\memorydumpR}{%
  \endnumberingR
  \numberingRtrue
  \global\pst@rtedRtrue
  \global\advance\section@numR \@ne
     \led@mess@SectionContinued{\the\section@numR R}%
  \line@list@stuffR{\jobname.\extensionchars\the\section@numR R}%
  \l@dend@stuff}

```

14 Line counting

14.1 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` `\bypstart@Rtrue` `\bypstart@Rfalse` `\ifbypage@R` `\bypage@Rtrue` `\bypage@Rfalse`

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

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

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

```
\newif\ifbypage@R
\newif\ifbypstart@R
  \bypage@Rfalse
  \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`.

```
\newcommand*{\lineationR}[1]{{%
  \ifnumbering
    \led@err@LineationInNumbered
  \else
    \def\@tempa{#1}\def\@tempb{page}%
    \ifx\@tempa\@tempb
        \global\bypage@Rtrue
        \global\bypstart@Rfalse
    \else
       \def\@tempb{pstart}%
       \ifx\@tempa\@tempb
           \global\bypage@Rfalse
           \global\bypstart@Rtrue
       \else
           \def@tempb{section}
           \ifx\@tempa\@tempb
           \global\bypage@Rfalse
           \global\bypstart@Rfalse
           \else
             \led@warn@BadLineation
        \fi
       \fi
    \fi
  \fi}}
```

`\lineation*` `\lineation*` change the lineation system for the side.

```
\WithSuffix\newcommand\lineation*[1]{%
  \lineation{#1}%
  \lineationR{#1}%
}%
```

`\linenummargin` `\line@marginR` 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.

```
\newcount\line@marginR
\renewcommand*{\linenummargin}[1]{{%
  \l@dgetline@margin{#1}%
  \ifnum\@l@dtempcntb>\m@ne
    \ifledRcol
      \global\line@marginR=\@l@dtempcntb
    \else
      \global\line@margin=\@l@dtempcntb
    \fi
  \fi}}
```

By default put right text numbers at the right.

```
\line@marginR=\@ne

```

`\c@firstlinenumR` `\c@linenumincrementR`

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.

```
\newcounter{firstlinenumR}
  \setcounter{firstlinenumR}{5}
\newcounter{linenumincrementR}
  \setcounter{linenumincrementR}{5}
```

`\c@firstsublinenumR` `\c@sublinenumincrementR`

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

```
\newcounter{firstsublinenumR}
  \setcounter{firstsublinenumR}{5}
\newcounter{sublinenumincrementR}
  \setcounter{sublinenumincrementR}{5}

```

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

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

```
\providecommand*{\firstlinenum}{}
\providecommand*{\linenumincrement}{}
\providecommand*{\firstsublinenum}{}
\providecommand*{\sublinenumincrement}{}
\renewcommand*{\firstlinenum}[1]{%
  \ifledRcol \setcounter{firstlinenumR}{#1}%
```

```
  \else      \setcounter{firstlinenum}{#1}%
  \fi}
\renewcommand*{\linenumincrement}[1]{%
  \ifledRcol \setcounter{linenumincrementR}{#1}%
  \else      \setcounter{linenumincrement}{#1}%
  \fi}
\renewcommand*{\firstsublinenum}[1]{%
  \ifledRcol \setcounter{firstsublinenumR}{#1}%
  \else      \setcounter{firstsublinenum}{#1}%
  \fi}
\renewcommand*{\sublinenumincrement}[1]{%
  \ifledRcol \setcounter{sublinenumincrementR}{#1}%
  \else      \setcounter{sublinenumincrement}{#1}%
  \fi}
\WithSuffix\newcommand\firstlinenum*[1]{\setcounter{firstlinenumR}{#1}\setcounter{firstlinenum}{#1}}
\WithSuffix\newcommand\linenumincrement*[1]{\setcounter{linenumincrementR}{#1}\setcounter{linenumincr
\WithSuffix\newcommand\firstsublinenum*[1]{\setcounter{subfirstlinenumR}{#1}\setcounter{subfirstlinen
\WithSuffix\newcommand\sublinenumincrement*[1]{\setcounter{sublinenumincrementR}{#1}\setcounter{subli
```

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

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

`\linenumrepR` `\sublinenumrepR` `\linenumrepR{`⟨*ctr*⟩`}` typesets the right line number ⟨*ctr*⟩, and similarly `\sublinenumrepR` for subline numbers.

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

`\leftlinenumR` `\rightlinenumR` `\l@dlinenumR` `\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`.

```
\newcommand*{\leftlinenumR}{%
  \l@dlinenumR
  \kern\linenumsep}
\newcommand*{\rightlinenumR}{%
  \kern\linenumsep
  \l@dlinenumR}
\newcommand*{\l@dlinenumR}{%
  \numlabfont\linenumrepR{\line@numR}\Rlineflag%
  \ifsublines@
    \ifnum\subline@num>\z@
      \unskip\fullstop\sublinenumrepR{\subline@numR}%
    \fi
  \fi}
```

14.2 Line-number counters and lists

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

`\line@numR` `\subline@numR` `\absline@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.

```
\newcount\line@numR
\newcount\subline@numR
\newcount\absline@numR
```

`\line@listR` `\insertlines@listR` `\actionlines@listR` `\actions@listR`

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

Here are the commands to create these lists:

```
\list@create{\line@listR}
\list@create{\insertlines@listR}
\list@create{\actionlines@listR}
\list@create{\actions@listR}
```

`\linesinpar@listL` `\linesinpar@listR` `\maxlinesinpar@list`

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

```
\list@create{\linesinpar@listL}
\list@create{\linesinpar@listR}
\list@create{\maxlinesinpar@list}
```

`\page@numR`

The right text page number.

```
\newcount\page@numR
```

14.3 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.

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

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

```
  \ifledRcol
    \list@clear{\line@listR}%
    \list@clear{\insertlines@listR}%
```

```
    \list@clear{\actionlines@listR}%
    \list@clear{\actions@listR}%
    \list@clear{\linesinpar@listR}%
    \list@clear{\linesonpage@listR}
  \else
    \list@clearing@reg
    \list@clear{\linesinpar@listL}%
    \list@clear{\linesonpage@listL}%
  \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).

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

Now get the file and interpret it.

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

```
  \ifledRcol
    \global\page@numR=\m@ne
    \ifx\actionlines@listR\empty
      \gdef\next@actionlineR{1000000}%
    \else
      \gl@p\actionlines@listR\to\next@actionlineR
      \gl@p\actions@listR\to\next@actionR
    \fi
  \else
    \global\page@num=\m@ne
    \ifx\actionlines@list\empty
      \gdef\next@actionline{1000000}%
    \else
      \gl@p\actionlines@list\to\next@actionline
      \gl@p\actions@list\to\next@action
    \fi
  \fi}

```

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.

14.4 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.

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

```
\newcommand{\@nl@regR}{%
  \ifx\l@dchset@num\relax \else
    \advance\absline@numR \@ne
    \set@line@action
    \let\l@dchset@num\relax
    \advance\absline@numR \m@ne
    \advance\line@numR \m@ne%   % do we need this?
  \fi
  \advance\absline@numR \@ne
  \ifx\next@page@numR\relax \else
    \page@action
    \let\next@page@numR\relax
  \fi
  \ifx\sub@change\relax \else
    \ifnum\sub@change>\z@
      \sublines@true
    \else
      \sublines@false
    \fi
    \sub@action
    \let\sub@change\relax
  \fi
  \ifcase\@lockR
  \or
    \@lockR \tw@
  \or\or
    \@lockR \z@
  \fi
  \ifcase\sub@lockR
  \or
    \sub@lockR \tw@
  \or\or
    \sub@lockR \z@
  \fi
  \ifsublines@
    \ifnum\sub@lockR<\tw@
      \advance\subline@numR \@ne
    \fi
  \else
    \ifnum\@lockR<\tw@
      \advance\line@numR \@ne \subline@numR \z@
    \fi
  \fi}

\renewcommand*{\@nl}[2]{%
```

```
  \fix@page{#1}%
  \ifledRcol
    \@nl@regR
  \else
    \@nl@reg
  \fi}
```

\last@page@numR
\fix@page

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

```
\newcount\last@page@numR
  \last@page@numR=-10000
\renewcommand*{\fix@page}[1]{%
  \ifledRcol
    \ifnum #1=\last@page@numR
    \else
      \ifbypage@R
        \line@numR \z@ \subline@numR \z@
      \fi
      \page@numR=#1\relax
      \last@page@numR=#1\relax
      \def\next@page@numR{#1}%
    \fi
  \else
    \ifnum #1=\last@page@num
    \else
      \ifbypage@
        \line@num \z@ \subline@num \z@
      \fi
      \page@num=#1\relax
      \last@page@num=#1\relax
      \def\next@page@num{#1}%
      \listcsxadd{normal@page@break}{\the\absline@num}
    \fi
  \fi}
```

\@adv

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

```
\renewcommand*{\@adv}[1]{%
  \ifsublines@
    \ifledRcol
      \advance\subline@numR by #1\relax
      \ifnum\subline@numR<\z@
        \led@warn@BadAdvancelineSubline
        \subline@numR \z@
      \fi
    \else
      \advance\subline@num by #1\relax
      \ifnum\subline@num<\z@
```

```
372       \led@warn@BadAdvancelineSubline
373       \subline@num \z@
374     \fi
375   \fi
376 \else
377   \ifledRcol
378     \advance\line@numR by #1\relax
379     \ifnum\line@numR<\z@
380       \led@warn@BadAdvancelineLine
381       \line@numR \z@
382     \fi
383   \else
384     \advance\line@num by #1\relax
385     \ifnum\line@num<\z@
386       \led@warn@BadAdvancelineLine
387       \line@num \z@
388     \fi
389   \fi
390 \fi
391 \set@line@action}
392
```

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

```
393 \renewcommand*{\@set}[1]{%
394   \ifledRcol
395     \ifsublines@
396       \subline@numR=#1\relax
397     \else
398       \line@numR=#1\relax
399     \fi
400     \set@line@action
401   \else
402     \ifsublines@
403       \subline@num=#1\relax
404     \else
405       \line@num=#1\relax
406     \fi
407     \set@line@action
408   \fi}
409
```

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

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

```
410 \renewcommand*{\l@d@set}[1]{%
411   \ifledRcol
412     \line@numR=#1\relax
```

```
    \advance\line@numR \@ne
    \def\l@dchset@num{#1}
  \else
    \line@num=#1\relax
    \advance\line@num \@ne
    \def\l@dchset@num{#1}
  \fi}
\let\l@dchset@num\relax
```

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

```
\renewcommand*{\page@action}{%
  \ifledRcol
    \xright@appenditem{\the\absline@numR}\to\actionlines@listR
    \xright@appenditem{\next@page@numR}\to\actions@listR
  \else
    \xright@appenditem{\the\absline@num}\to\actionlines@list
    \xright@appenditem{\next@page@num}\to\actions@list
  \fi}
```

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

```
\renewcommand*{\set@line@action}{%
  \ifledRcol
    \xright@appenditem{\the\absline@numR}\to\actionlines@listR
    \ifsublines@
       \@l@dtempcnta=-\subline@numR
    \else
       \@l@dtempcnta=-\line@numR
    \fi
    \advance\@l@dtempcnta by -5000\relax
    \xright@appenditem{\the\@l@dtempcnta}\to\actions@listR
  \else
    \xright@appenditem{\the\absline@num}\to\actionlines@list
    \ifsublines@
       \@l@dtempcnta=-\subline@num
    \else
       \@l@dtempcnta=-\line@num
    \fi
    \advance\@l@dtempcnta by -5000\relax
    \xright@appenditem{\the\@l@dtempcnta}\to\actions@list
  \fi}
```

`\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.

```
\renewcommand*{\sub@action}{%
  \ifledRcol
    \xright@appenditem{\the\absline@numR}\to\actionlines@listR
```

```
    \ifsublines@
      \xright@appenditem{-1001}\to\actions@listR
    \else
      \xright@appenditem{-1002}\to\actions@listR
    \fi
  \else
    \xright@appenditem{\the\absline@num}\to\actionlines@list
    \ifsublines@
      \xright@appenditem{-1001}\to\actions@list
    \else
      \xright@appenditem{-1002}\to\actions@list
    \fi
  \fi}
```

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

```
\newcount\@lockR
\newcount\sub@lockR

\newcommand*{\do@lockonR}{%
  \xright@appenditem{\the\absline@numR}\to\actionlines@listR
  \ifsublines@
    \xright@appenditem{-1005}\to\actions@listR
    \ifnum\sub@lockR=\z@
      \sub@lockR \@ne
    \else
      \ifnum\sub@lockR=\thr@@
        \sub@lockR \@ne
      \fi
    \fi
  \else
    \xright@appenditem{-1003}\to\actions@listR
    \ifnum\@lockR=\z@
      \@lockR \@ne
    \else
      \ifnum\@lockR=\thr@@
        \@lockR \@ne
      \fi
    \fi
  \fi}

\renewcommand*{\do@lockon}{%
  \ifx\next\lock@off
    \global\let\lock@off=\skip@lockoff
  \else
    \ifledRcol
      \do@lockonR
    \else
```

```
      \do@lockonL
    \fi
  \fi}
```

\lock@off \lock@off adds an entry to the action-code list to turn line number locking off.

\do@lockoff
\do@lockoffR
\skip@lockoff

```
\newcommand{\do@lockoffR}{%
  \xright@appenditem{\the\absline@numR}\to\actionlines@listR
  \ifsublines@
    \xright@appenditem{-1006}\to\actions@listR
    \ifnum\sub@lockR=\tw@
      \sub@lockR \thr@@
    \else
      \sub@lockR \z@
    \fi
  \else
    \xright@appenditem{-1004}\to\actions@listR
    \ifnum\@lockR=\tw@
      \@lockR \thr@@
    \else
      \@lockR \z@
    \fi
  \fi}

\renewcommand*{\do@lockoff}{%
  \ifledRcol
    \do@lockoffR
  \else
    \do@lockoffL
  \fi}
\global\let\lock@off=\do@lockoff

```

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

```
\providecommand*{\n@num}{}
\renewcommand*{\n@num}{%
  \ifledRcol
    \xright@appenditem{\the\absline@numR}\to\actionlines@listR
    \xright@appenditem{-1007}\to\actions@listR
  \else
    \n@num@reg
  \fi}

```

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

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

```
    \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.

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

```
  \begingroup
    \let\@ref=\dummy@ref
    \let\page@action=\relax
    \let\sub@action=\relax
    \let\set@line@action=\relax
    \let\@lab=\relax
    #2
    \global\endpage@num=\page@numR
    \global\endline@num=\line@numR
    \global\endsubline@num=\subline@numR
  \endgroup
```

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

```
    \xright@appenditem%
      {\the\page@numR|\the\line@numR|%
       \ifsublines@ \the\subline@numR \else 0\fi|%
       \the\endpage@num|\the\endline@num|%
       \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.

```
  #2
  \else
```

And when not in right text

```
    \@ref@reg{#1}{#2}%
  \fi}
```

`\@pend` `\@pendR` `\@pend{`⟨*num*⟩`}` adds its argument to the `\linesinpar@listL` list, and analagously 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.

```
\providecommand*{\@pend}[1]{}
\renewcommand*{\@pend}[1]{%
  \ifbypstart@\global\line@num=0\fi%
  \xright@appenditem{#1}\to\linesinpar@listL}
\providecommand*{\@pendR}[1]{}
\renewcommand*{\@pendR}[1]{%
  \ifbypstart@R\global\line@numR=0\fi
  \xright@appenditem{#1}\to\linesinpar@listR}

```

`\@lopL` `\@lopR` `\@lopL{`⟨*num*⟩`}` adds its argument to the `\linesonpage@listL` list, and analagously 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.

```
\providecommand*{\@lopL}[1]{}
\renewcommand*{\@lopL}[1]{%
  \xright@appenditem{#1}\to\linesonpage@listL}
\providecommand*{\@lopR}[1]{}
\renewcommand*{\@lopR}[1]{%
  \xright@appenditem{#1}\to\linesonpage@listR}

```

14.5 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`.

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

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

```
\newif\iffirst@linenum@out@R
  \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.

```
\newcommand*{\line@list@stuffR}[1]{%
  \read@linelist{#1}%
```

```
\iffirst@linenum@out@R
   \immediate\closeout\linenum@outR
   \global\first@linenum@out@Rfalse
   \immediate\openout\linenum@outR=#1
\else
  \if@minipage%
    \if@ledgroup%
      \closeout\linenum@outR
      \openout\linenum@outR=#1
    \else
      \immediate\closeout\linenum@outR
      \immediate\openout\linenum@outR=#1\relax
    \fi
  \else
     \closeout\linenum@outR%
     \openout\linenum@outR=#1\relax%
  \fi
\fi}

```

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

```
\newcommand*{\new@lineL}{%
  \write\linenum@out{\string\@nl[\the\c@page][\thepage]}}
```

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

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

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

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

```
\renewcommand*{\startsub}{\dimen0\lastskip
  \ifdim\dimen0>0pt \unskip \fi
  \ifledRcol \write\linenum@outR{\string\sub@on}%
  \else      \write\linenum@out{\string\sub@on}%
  \fi
  \ifdim\dimen0>0pt \hskip\dimen0 \fi}
\def\endsub{\dimen0\lastskip
  \ifdim\dimen0>0pt \unskip \fi
  \ifledRcol \write\linenum@outR{\string\sub@off}%
  \else      \write\linenum@out{\string\sub@off}%
  \fi
  \ifdim\dimen0>0pt \hskip\dimen0 \fi}

```

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

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

```
\renewcommand*{\setline}[1]{%
  \ifnum#1<\z@
    \led@warn@BadSetline
  \else
    \ifledRcol \write\linenum@outR{\string\@set[#1]}%
    \else      \write\linenum@out{\string\@set[#1]}%
    \fi
  \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.

```
\renewcommand*{\setlinenum}[1]{%
  \ifnum#1<\z@
    \led@warn@BadSetlinenum
  \else
    \ifledRcol \write\linenum@outR{\string\l@d@set[#1]}
    \else      \write\linenum@out{\string\l@d@set[#1]} \fi
  \fi}

```

`\startlock` `\endlock` 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.

```
\renewcommand*{\startlock}{%
  \ifledRcol \write\linenum@outR{\string\lock@on}%
  \else      \write\linenum@out{\string\lock@on}%
  \fi}
\def\endlock{%
  \ifledRcol \write\linenum@outR{\string\lock@off}%
  \else      \write\linenum@out{\string\lock@off}%
  \fi}

```

`\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.

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

```
\else
  \skipnumbering@reg
\fi}
```

15 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.

```
\long\def\critext#1#2/{\leavevmode
  \begingroup
    \renewcommand{\@tag}{\no@expands #1}%
    \set@line
    \ifledRcol \global\insert@countR \z@
    \else      \global\insert@count \z@ \fi
    \ignorespaces #2\relax
    \@ifundefined{xpg@main@language}{%if not polyglossia
       \flag@start}%
       {\if@RTL\flag@end\else\flag@start\fi% be careful on the direction of writing with p
       }%
  \endgroup
  \showlemma{#1}%
  \ifx\end@lemmas\empty \else
    \gl@p\end@lemmas\to\x@lemma
    \x@lemma
    \global\let\x@lemma=\relax
  \fi
  \@ifundefined{xpg@main@language}{%if not polyglossia
       \flag@end}%
       {\if@RTL\flag@start\else\flag@end\fi% be careful on the direction of writing with p
       }
  }
```

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

```
\renewcommand{\edtext}[2]{\leavevmode
  \begingroup%
    \renewcommand{\@tag}{\no@expands #1}%
    \set@line%
    \ifledRcol \global\insert@countR \z@%
    \else      \global\insert@count \z@ \fi%
    \ignorespaces #2\relax%
    \@ifundefined{xpg@main@language}{%if not polyglossia
       \flag@start}%
       {\if@RTL\flag@end\else\flag@start\fi% be careful on the direction of writing with polyglossia
       }%
  \endgroup%
  \showlemma{#1}%
  \ifx\end@lemmas\empty \else%
    \gl@p\end@lemmas\to\x@lemma%
    \x@lemma%
    \global\let\x@lemma=\relax%
  \fi%
  \@ifundefined{xpg@main@language}{%if not polyglossia
       \flag@end}%
       {\if@RTL\flag@start\else\flag@end\fi% be careful on the direction of writing with polyglossia
       }%
}

```

`\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`.

```
\renewcommand*{\set@line}{%
  \ifledRcol
    \ifx\line@listR\empty
      \global\noteschanged@true
      \xdef\l@d@nums{000|000|000|000|000|000|\edfont@info}%
    \else
      \gl@p\line@listR\to\@tempb
      \xdef\l@d@nums{\@tempb|\edfont@info}%
      \global\let\@tempb=\undefined
    \fi
  \else
    \ifx\line@list\empty
      \global\noteschanged@true
      \xdef\l@d@nums{000|000|000|000|000|000|\edfont@info}%
    \else
      \gl@p\line@list\to\@tempb
      \xdef\l@d@nums{\@tempb|\edfont@info}%
      \global\let\@tempb=\undefined
    \fi
  \fi}

```

16 Parallel environments

The initial set up for parallel processing is deceptively simple.

pairs
pages
chapterinpages

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

```
\newenvironment{pairs}{%
  \l@dpairingtrue
  \l@dpagingfalse
  \initnumbering@sectcmd
}{%
  \l@dpairingfalse
}
```

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.

```
\newenvironment{pages}{%
 \let\oldchapter\chapter
 \let\chapter\chapterinpages
  \l@dpairingtrue
  \l@dpagingtrue
  \initnumbering@sectcmd
  \setlength{\Lcolwidth}{\textwidth}%
  \setlength{\Rcolwidth}{\textwidth}%
}{%
  \l@dpairingfalse
  \l@dpagingfalse
  \let\chapter\oldchapter
}
\newcommand{\chapterinpages}{\thispagestyle{plain}%
                     \global\@topnum\z@
                     \@afterindentfalse
                     \secdef\@chapter\@schapter}

```

ifinstanzaL
ifinstanzaR

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

```
\newif\ifinstanzaL
\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.

```
\newenvironment{Leftside}{%
  \ledRcolfalse
  \setcounter{pstartL}{1}
```

```
  \let\pstart\pstartL
  \let\thepstart\thepstartL
  \let\pend\pendL
  \let\memorydump\memorydumpL
  \Leftsidehook
  \let\old@startstanza\@startstanza
  \def\@startstanza[##1]{\global\instanzaLtrue\old@startstanza[##1]}
}{
    \Leftsidehookend}
```

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

`\Rightsidehook` `\Rightsidehookend`

```
\newcommand*{\Leftsidehook}{}
\newcommand*{\Leftsidehookend}{}
\newcommand*{\Rightsidehook}{}
\newcommand*{\Rightsidehookend}{}
```

`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.

```
\newenvironment{Rightside}{%
  \ledRcoltrue
  \let\beginnumbering\beginnumberingR
  \let\endnumbering\endnumberingR
  \let\pausenumbering\pausenumberingR
  \let\resumenumbering\resumenumberingR
  \let\memorydump\memorydumpR
  \let\thepstart\thepstartR
  \let\pstart\pstartR
  \let\pend\pendR
  \let\ledpb\ledpbR
  \let\lednopb\lednopbR
  \let\lineation\lineationR
  \Rightsidehook
  \let\old@startstanza\@startstanza
  \def\@startstanza[##1]{\global\instanzaRtrue\old@startstanza[##1]}
}{%
  \ledRcolfalse
  \Rightsidehookend
}
```

17 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.

17.1 Boxes, counters, \pstart and \pend

\num@linesR
\one@lineR
\par@lineR

Here are numbers and flags that are used internally in the course of the paragraph decomposition.

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.

```
\newcount\num@linesR
\newbox\one@lineR
\newcount\par@lineR
```

\pstartL
\pstartR

\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.

```

\newcounter{pstartL}
\newcounter{pstartLold}
\renewcommand{\thepstartL}{{\bfseries\@arabic\c@pstartL}. }
\newcounter{pstartR}
\newcounter{pstartRold}
\renewcommand{\thepstartR}{{\bfseries\@arabic\c@pstartR}. }

\newcommandx*{\pstartL}[1][1]{%
  \if@nobreak%
    \let\@oldnobreak\@nobreaktrue%
  \else%
    \let\@oldnobreak\@nobreakfalse%
  \fi%
    \@nobreaktrue%
  \ifnumbering \else%
    \led@err@PstartNotNumbered%
```

```
    \beginnumbering%
  \fi%
  \ifnumberedpar@%
    \led@err@PstartInPstart%
    \pend%
  \fi%
```

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

```
  \ifpst@rtedL\else%
      \setcounter{pstartLold}{\value{pstartL}}%
    \list@clear{\inserts@list}%
    \global\let\next@insert=\empty%
     \global\pst@rtedLtrue%
  \fi%
  \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.

```
  \global\advance\l@dnumpstartsL \@ne%
  \ifnum\l@dnumpstartsL>\l@dc@maxchunks%
    \led@err@TooManyPstarts%
    \global\l@dnumpstartsL=\l@dc@maxchunks%
  \fi%
  \global\setnamebox{l@dLcolrawbox\the\l@dnumpstartsL}=\vbox\bgroup%
  \ifautopar\else%
    \ifnumberpstart%
      \ifsidepstartnum%
        \else%
         \thepstartL%
        \fi%
       \fi%
      \fi%
  \hsize=\Lcolwidth%
  \numberedpar@true%
  \iflabelpstart\protected@edef\@currentlabel%
       {\p@pstartL\thepstartL}\fi%
```

Dump the optional arguments

```
  \ifstrempty{#1}%
    {}%
    {\csgdef{before@pstartL@\the\l@dnumpstartsL}{\noindent#1}}%
  }
\newcommandx*{\pstartR}[1][1]{%
  \if@nobreak%
    \let\@oldnobreak\@nobreaktrue%
  \else%
    \let\@oldnobreak\@nobreakfalse%
  \fi%
    \@nobreaktrue%
```

```
\ifnumberingR \else%
  \led@err@PstartNotNumbered%
  \beginnumberingR%
\fi%
\ifnumberedpar@%
  \led@err@PstartInPstart%
  \pendR%
\fi%
\ifpst@rtedR\else%
    \setcounter{pstartRold}{\value{pstartR}}%
  \list@clear{\inserts@listR}%
  \global\let\next@insertR=\empty%
  \global\pst@rtedRtrue%
\fi%
\begingroup\normal@pars%
\global\advance\l@dnumpstartsR \@ne%
\ifnum\l@dnumpstartsR>\l@dc@maxchunks%
  \led@err@TooManyPstarts%
  \global\l@dnumpstartsR=\l@dc@maxchunks%
\fi%
\global\setnamebox{l@dRcolrawbox\the\l@dnumpstartsR}=\vbox\bgroup%
  \ifautopar\else%
    \ifnumberpstart%
      \ifsidepstartnum\else%
        \thepstartR%
       \fi%
      \fi%
     \fi%
\hsize=\Rcolwidth%
\numberedpar@true%
 \iflabelpstart\protected@edef\@currentlabel%
     {\p@pstartR\thepstartR}\fi%
 \ifstrempty{#1}%
   {}
   {\csgdef{before@pstartR@\the\l@dnumpstartsR}{\noindent#1}}%
}
```

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

```
\newcommandx*{\pendL}[1][1]{%
  \ifnumbering \else%
    \led@err@PendNotNumbered%
  \fi%
  \ifnumberedpar@ \else%
    \led@err@PendNoPstart%
  \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.

```
\l@dzeropenalties%
\endgraf\global\num@lines=\prevgraf\egroup%
\global\par@line=0%
```

End the group that was begun in the \pstart.

```
\endgroup%
\ignorespaces%
\@oldnobreak%
\ifnumberpstart%
  \addtocounter{pstartL}{1}%
\fi
\parledgroup@beforenotes@save{L}%
```

Dump content of the optional argument.

```
\ifstrempty{#1}%
  {}%
  {\csgdef{after@pendL@\the\l@dnumpstartsL}{\noindent#1}}%
}
```

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

```
\newcommandx*{\pendR}[1][1]{%
  \ifnumberingR \else%
    \led@err@PendNotNumbered%
  \fi%
  \ifnumberedpar@ \else%
    \led@err@PendNoPstart%
  \fi%
  \l@dzeropenalties%
  \endgraf\global\num@linesR=\prevgraf\egroup%
  \global\par@lineR=0%
  \endgroup%
  \ignorespaces%
  \@oldnobreak%
  \ifnumberpstart%
    \addtocounter{pstartR}{1}%
  \fi%
  \parledgroup@beforenotes@save{R}%
  \ifstrempty{#1}%
    {}%
    {\csgdef{after@pendR@\the\l@dnumpstartsR}{\noindent#1}}%
}

```

17.2 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
\l@drightbox

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

```
\newbox\l@dleftbox
\newbox\l@drightbox
```

\countLline
\countRline

We need to know the number of lines processed.

```
\newcount\countLline
  \countLline \z@
\newcount\countRline
  \countRline \z@
```

\@donereallinesL
\@donetotallinesL
\@donereallinesR
\@donetotallinesR

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

```
\newcount\@donereallinesL
\newcount\@donetotallinesL
\newcount\@donereallinesR
\newcount\@donetotallinesR
```

\do@lineL

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

```
\newcommand*{\do@lineL}{%
  \advance\countLline \@ne
  \ifvbox\namebox{l@dLcolrawbox\the\l@dpscL}%
  {\vbadness=10000
   \splittopskip=\z@
   \do@lineLhook
   \l@demptyd@ta
   \global\setbox\one@line=\vsplit\namebox{l@dLcolrawbox\the\l@dpscL}
                           to\baselineskip}%
  \IfStrEq{\splitfirstmarks\parledgroup@}{begin}{\parledgroup@notes@startL}{}
  \unvbox\one@line \global\setbox\one@line=\lastbox
  \getline@numL
  \ifnum\@lock>\@ne%
    \inserthangingsymboltrue%
  \else%
    \inserthangingsymbolfalse%
  \fi
  \setbox\l@dleftbox
  \hb@xt@ \Lcolwidth{%
    \affixline@num
    \xifinlist{\the\l@dpscL}{\eled@sections@@}%
      {}%
      {\print@lineL}}%
  \add@penaltiesL
```

```
  \global\advance\@donereallinesL\@ne
  \global\advance\@donetotallinesL\@ne
\else
  \setbox\l@dleftbox \hb@xt@ \Lcolwidth{\hspace*{\Lcolwidth}}%
  \global\advance\@donetotallinesL\@ne
\fi}
```

`\print@eledsectionL` `\print@lineL` is for lines without a sectioning command.

```
\def\print@lineL{%
    \affixpstart@numL%
    \l@dld@ta %space kept for backward compatibility
    \add@inserts\affixside@note%
    \l@dlsn@te %space kept for backward compatibility
    {\ledllfill\hb@xt@ \wd\one@line{\do@insidelineLhook\inserthangingsymbolL\new@lineL\l@dunhbox@line
     \l@drsn@te}}
```

`\print@eledsectionL` `\print@eledsectionL` is for line with macro code.

```
\def\print@eledsectionL{%
    \add@inserts\affixside@note%
    \addtocounter{pstartL}{-1}%
    \ifdefstring{\@eledsectnotoc}{L}{\ledsectnotoc}{}
    \ifdefstring{\@eledsectmark}{L}{}{\ledsectnomark}
    \numdef{\temp@}{\l@dpscL-1}%
    \xifinlist{\temp@}{\eled@sections@@}{\@nobreaktrue}{\@nobreakfalse}%
    \@eled@sectioningtrue%
    \csuse{eled@sectioning@\the\l@dpscL}%
    \@eled@sectioningfalse%
    \global\csundef{eled@sectioning@\the\l@dpscL}%
    \if@RTL%
      \hspace{-3\paperwidth}%
    {\hbox{\l@dunhbox@line{\one@line}} \new@line}%
    \else%
      \hspace{3\paperwidth}%
    {\new@line \hbox{\l@dunhbox@line{\one@line}}}%
    \fi%
    \vskip\eledsection@correcting@skip%
}
```

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

```
\newcommand*{\do@lineLhook}{}
\newcommand*{\do@lineRhook}{}
\newcommand*{\do@insidelineLhook}{}
\newcommand*{\do@insidelineRhook}{}
```

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

```
\newcommand*{\do@lineR}{%
  \ledRcol@true%
  \advance\countRline \@ne
  \ifvbox\namebox{l@dRcolrawbox\the\l@dpscR}%
  {\vbadness=10000
   \splittopskip=\z@
   \do@lineRhook
   \l@demptyd@ta
   \global\setbox\one@lineR=\vsplit\namebox{l@dRcolrawbox\the\l@dpscR}
                              to\baselineskip}%
  \IfStrEq{\splitfirstmarks\parledgroup@}{begin}{\parledgroup@notes@startR}{}
  \unvbox\one@lineR \global\setbox\one@lineR=\lastbox
  \getline@numR
  \ifnum\@lockR>\@ne%
    \inserthangingsymbolRtrue
  \else%
    \inserthangingsymbolRfalse%
  \fi%
  \setbox\l@drightbox
  \hb@xt@ \Rcolwidth{%
    \affixline@numR%
    \xifinlist{\the\l@dpscR}{\eled@sectionsR@@}%
     {}%
     {\print@lineR}%
  }%
  \add@penaltiesR
  \global\advance\@donereallinesR\@ne
  \global\advance\@donetotallinesR\@ne
\else
  \setbox\l@drightbox \hb@xt@ \Rcolwidth{\hspace*{\Rcolwidth}}
  \global\advance\@donetotallinesR\@ne
\fi
\ledRcol@false%
}
```

\print@lineR
\print@eledsectionR

17.3 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.

```
\newcommand*{\getline@numR}{%
  \global\advance\absline@numR \@ne
  \do@actionsR
  \do@ballastR
\ifledgroupnotesR@\else\ifnumberline
  \ifsublines@
```

```
    \ifnum\sub@lockR<\tw@
      \global\advance\subline@numR \@ne
    \fi
  \else
    \ifnum\@lockR<\tw@
      \global\advance\line@numR \@ne
      \global\subline@numR \z@
    \fi
  \fi
\fi
\fi
}
\newcommand*{\getline@numL}{%
    \global\advance\absline@num \@ne
  \do@actions
  \do@ballast
\ifledgroupnotesL@\else\ifnumberline
  \ifsublines@
    \ifnum\sub@lock<\tw@
      \global\advance\subline@num \@ne
    \fi
  \else
    \ifnum\@lock<\tw@
      \global\advance\line@num \@ne
      \global\subline@num \z@
    \fi
  \fi
\fi
\fi
}


```

`\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.

```
\newcommand*{\do@ballastR}{\global\ballast@count=\z@
  \begingroup
    \advance\absline@numR \@ne
    \ifnum\next@actionlineR=\absline@numR
      \ifnum\next@actionR>-1001
        \global\advance\ballast@count by -\c@ballast
      \fi
    \fi
  \endgroup}
```

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

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

```
\newcommand*{\do@actions@fixedcodeR}{%
  \ifcase\@l@dtempcnta%
  \or%                      % 1001
    \global\sublines@true
  \or%                      % 1002
    \global\sublines@false
  \or%                      % 1003
    \global\@lockR=\@ne
  \or%                      % 1004
    \ifnum\@lockR=\tw@
      \global\@lockR=\thr@@
    \else
      \global\@lockR=\z@
    \fi
  \or%                      % 1005
     \global\sub@lockR=\@ne
  \or%                      % 1006
    \ifnum\sub@lockR=\tw@
      \global\sub@lockR=\thr@@
    \else
      \global\sub@lockR=\z@
    \fi
  \or%                      % 1007
    \l@dskipnumbertrue
  \else
    \led@warn@BadAction
  \fi}


\newcommand*{\do@actionsR}{%
  \global\let\do@actions@nextR=\relax
  \@l@dtempcntb=\absline@numR
  \ifnum\@l@dtempcntb<\next@actionlineR\else
    \ifnum\next@actionR>-1001\relax
      \global\page@numR=\next@actionR
      \ifbypage@R
        \global\line@numR \z@  \global\subline@numR \z@
      \fi
    \else
      \ifnum\next@actionR<-4999\relax   %  9/05 added relax here
        \@l@dtempcnta=-\next@actionR
        \advance\@l@dtempcnta by -5001\relax
        \ifsublines@
          \global\subline@numR=\@l@dtempcnta
        \else
          \global\line@numR=\@l@dtempcnta
        \fi
      \else
        \@l@dtempcnta=-\next@actionR
        \advance\@l@dtempcnta by -1000\relax
```

```
        \do@actions@fixedcodeR
      \fi
    \fi
    \ifx\actionlines@listR\empty
      \gdef\next@actionlineR{1000000}%
    \else
      \gl@p\actionlines@listR\to\next@actionlineR
      \gl@p\actions@listR\to\next@actionR
      \global\let\do@actions@nextR=\do@actionsR
    \fi
  \fi
  \do@actions@nextR}
```

17.4 Line number printing

\l@dcalcnum
\ch@cksub@l@ckR
\ch@ck@l@ckR
\f@x@l@cksR
\affixline@numR

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

```
\providecommand*{\l@dcalcnum}[3]{%
  \ifnum #1 > #2\relax
    \@l@dtempcnta = #1\relax
    \advance\@l@dtempcnta by -#2\relax
    \divide\@l@dtempcnta by #3\relax
    \multiply\@l@dtempcnta by #3\relax
    \advance\@l@dtempcnta by #2\relax
  \else
    \@l@dtempcnta=#2\relax
  \fi}

\newcommand*{\ch@cksub@l@ckR}{%
  \ifcase\sub@lockR
  \or
    \ifnum\sublock@disp=\@ne
      \@l@dtempcntb \z@  \@l@dtempcnta \@ne
    \fi
  \or
    \ifnum\sublock@disp=\tw@
    \else
      \@l@dtempcntb \z@  \@l@dtempcnta \@ne
    \fi
  \or
    \ifnum\sublock@disp=\z@
      \@l@dtempcntb \z@  \@l@dtempcnta \@ne
    \fi
  \fi}

\newcommand*{\ch@ck@l@ckR}{%
  \ifcase\@lockR
  \or
```

```
    \ifnum\lock@disp=\@ne
      \@l@dtempcntb \z@  \@l@dtempcnta \@ne
    \fi
  \or
    \ifnum\lock@disp=\tw@
    \else
      \@l@dtempcntb \z@  \@l@dtempcnta \@ne
    \fi
  \or
    \ifnum\lock@disp=\z@
      \@l@dtempcntb \z@  \@l@dtempcnta \@ne
    \fi
  \fi}

\newcommand*{\f@x@l@cksR}{%
  \ifcase\@lockR
  \or
    \global\@lockR \tw@
  \or \or
    \global\@lockR \z@
  \fi
  \ifcase\sub@lockR
  \or
    \global\sub@lockR \tw@
  \or \or
    \global\sub@lockR \z@
  \fi}


\newcommand*{\affixline@numR}{%
\ifledgroupnotesR@\else\ifnumberline
\ifl@dskipnumber
  \global\l@dskipnumberfalse
\else
  \ifsublines@
    \@l@dtempcntb=\subline@numR
    \l@dcalcnum{\subline@numR}{\c@firstsublinenumR}{\c@sublinenumincrementR}%
    \ch@cksub@lockR
  \else
    \@l@dtempcntb=\line@numR
    \ifx\linenumberlist\empty
      \l@dcalcnum{\line@numR}{\c@firstlinenumR}{\c@linenumincrementR}%
    \else
      \@l@dtempcnta=\line@numR
      \edef\rem@inder{,\linenumberlist,\number\line@numR,}%
        \edef\sc@n@list{\def\noexpand\sc@n@list
        ####1,\number\@l@dtempcnta,####2|{\def\noexpand\rem@inder{####2}}}%
      \sc@n@list\expandafter\sc@n@list\rem@inder|%
        \ifx\rem@inder\empty\advance\@l@dtempcnta\@ne\fi
     \fi
```

```
    \ch@ck@l@ckR
  \fi
  \ifnum\@l@dtempcnta=\@l@dtempcntb
    \if@twocolumn
      \if@firstcolumn
        \gdef\l@dld@ta{\llap{{\leftlinenumR}}}%
      \else
        \gdef\l@drd@ta{\rlap{{\rightlinenumR}}}%
      \fi
    \else
      \@l@dtempcntb=\line@marginR
      \ifnum\@l@dtempcntb>\@ne
        \advance\@l@dtempcntb by\page@numR
      \fi
      \ifodd\@l@dtempcntb
        \gdef\l@drd@ta{\rlap{{\rightlinenumR}}}%
      \else
        \gdef\l@dld@ta{\llap{{\leftlinenumR}}}%
      \fi
    \fi
  \fi
  \f@x@l@cksR
\fi
\fi
\fi}
```

17.5 Pstart number printing in side

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

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

`\affixpstart@numL`
`\affixpstart@numR`
`\leftpstartnumR`
`\rightpstartnumR`
`\leftpstartnumL`
`\rightpstartnumL`
`\ifpstartnumR`

```

\newcommand*{\affixpstart@numL}{%
\ifsidepstartnum
\if@twocolumn
    \if@firstcolumn
      \gdef\l@dld@ta{\llap{{\leftpstartnumL}}}%
    \else
      \gdef\l@drd@ta{\rlap{{\rightpstartnumL}}}%
    \fi
  \else
    \@l@dtempcntb=\line@margin
    \ifnum\@l@dtempcntb>\@ne
      \advance\@l@dtempcntb \page@num
```

```
    \fi
    \ifodd\@l@dtempcntb
      \gdef\l@drd@ta{\rlap{{\rightpstartnumL}}}%
    \else
      \gdef\l@dld@ta{\llap{{\leftpstartnumL}}}%
    \fi
  \fi
\fi
}
\newcommand*{\affixpstart@numR}{%
\ifsidepstartnum
\if@twocolumn
    \if@firstcolumn
      \gdef\l@dld@ta{\llap{{\leftpstartnumR}}}%
    \else
      \gdef\l@drd@ta{\rlap{{\rightpstartnumR}}}%
    \fi
  \else
    \@l@dtempcntb=\line@marginR
    \ifnum\@l@dtempcntb>\@ne
      \advance\@l@dtempcntb \page@numR
    \fi
    \ifodd\@l@dtempcntb
      \gdef\l@drd@ta{\rlap{{\rightpstartnumR}}}%
    \else
      \gdef\l@dld@ta{\llap{{\leftpstartnumR}}}%
    \fi
  \fi
\fi
}

\newcommand*{\leftpstartnumL}{
\ifpstartnum
\thepstartL
\kern\linenumsep\global\pstartnumfalse\fi
}
\newcommand*{\rightpstartnumL}{
\ifpstartnum\kern\linenumsep
\thepstartL
\global\pstartnumfalse\fi
}
\newif\ifpstartnumR
\pstartnumRtrue
\newcommand*{\leftpstartnumR}{
\ifpstartnumR
\thepstartR
\kern\linenumsep\global\pstartnumRfalse\fi
}
\newcommand*{\rightpstartnumR}{
\ifpstartnumR\kern\linenumsep
```

```
\thepstartR
\global\pstartnumRfalse\fi
}
```

17.6 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.

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

\add@insertsR
\add@inserts@nextR

The right text version.

```
\newcommand*{\add@insertsR}{%
  \global\let\add@inserts@nextR=\relax
  \ifx\inserts@listR\empty \else
    \ifx\next@insertR\empty
      \ifx\insertlines@listR\empty
        \global\noteschanged@true
        \gdef\next@insertR{100000}%
      \else
        \gl@p\insertlines@listR\to\next@insertR
      \fi
    \fi
    \ifnum\next@insertR=\absline@numR
      \gl@p\inserts@listR\to\@insertR
      \@insertR
      \global\let\@insertR=\undefined
      \global\let\next@insertR=\empty
      \global\let\add@inserts@nextR=\add@insertsR
    \fi
  \fi
  \add@inserts@nextR}

```

17.7 Penalties

\add@penaltiesL
\add@penaltiesR

\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@dtempcnta 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@dtempcnta=\ballast@count
```

```
\ifnum\num@linesR>\@ne
  \global\advance\par@lineR \@ne
  \ifnum\par@lineR=\@ne
    \advance\@l@dtempcnta by \clubpenalty
  \fi
  \@l@dtempcntb=\par@lineR \advance\@l@dtempcntb \@ne
  \ifnum\@l@dtempcntb=\num@linesR
    \advance\@l@dtempcnta by \widowpenalty
  \fi
  \ifnum\par@lineR<\num@linesR
    \advance\@l@dtempcnta by \interlinepenalty
  \fi
\fi
  \ifnum\@l@dtempcnta=\z@
    \relax
  \else
    \ifnum\@l@dtempcnta>-10000
      \penalty\@l@dtempcnta
    \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 nor 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.

```
\newcommand*{\add@penaltiesL}{}
\newcommand*{\add@penaltiesR}{}
```

17.8 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.

```
\newcommand*{\flush@notesR}{%
  \@xloop
    \ifx\inserts@listR\empty \else
      \gl@p\inserts@listR\to\@insertR
      \@insertR
      \global\let\@insertR=\undefined
  \repeat}
```

18 Footnotes

18.1 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
```

```
\def\printlinesR#1|#2|#3|#4|#5|#6|#7|{\begingroup
  \setprintlines{#1}{#2}{#3}{#4}{#5}{#6}%
  \ifl@d@pnum #1\fullstop\fi
  \ifledplinenum \linenumr@p{#2}\Rlineflag\else \symplinenum\fi
  \ifl@d@ssub \fullstop \sublinenumr@p{#3}\fi
  \ifl@d@dash \endashchar\fi
  \ifl@d@pnum #4\fullstop\fi
  \ifl@d@elin \linenumr@p{#5}\Rlineflag\fi
  \ifl@d@esl \ifl@d@elin \fullstop\fi \sublinenumr@p{#6}\fi
\endgroup}

\let\ledsavedprintlines\printlines

```

19 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.

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

```

`\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.

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

```
      \gl@p\labelref@listR\to\label@refs
    \fi
    \ifvmode
        \advancelabel@refs
    \fi
    \protected@write\@auxout{}%
      {\string\l@dmake@labelsR\space\thepage|\label@refs|\the\c@pstartR|{#1}}%
  \else
    \write\linenum@out{\string\@lab}%
    \ifx\labelref@list\empty
      \xdef\label@refs{\zz@@@}%
    \else
      \gl@p\labelref@list\to\label@refs
    \fi
    \ifvmode
        \advancelabel@refs
    \fi
  \protected@write\@auxout{}%
    {\string\l@dmake@labels\space\thepage|\label@refs|\the\c@pstart|{#1}}%
  \fi
  \@esphack}
```

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

```
\def\l@dmake@labelsR#1|#2|#3|#4|#5{%
  \expandafter\ifx\csname the@label#5\endcsname \relax\else
    \led@warn@DuplicateLabel{#4}%
  \fi
  \expandafter\gdef\csname the@label#5\endcsname{#1|#2\Rlineflag|#3|#4}%
  \ignorespaces}
\AtBeginDocument{%
  \def\l@dmake@labelsR#1|#2|#3|#4|#5{}%
}
```

`\@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`, `\@nl`, and the `\sub@on` and `\sub@off` commands appearing in the `\linenum@out` file.

```
\renewcommand*{\@lab}{%
  \ifledRcol
    \xright@appenditem{\linenumr@p{\line@numR}|%
      \ifsublines@ \sublinenumr@p{\subline@numR}\else 0\fi}%
      \to\labelref@listR
  \else
    \xright@appenditem{\linenumr@p{\line@num}|%
      \ifsublines@ \sublinenumr@p{\subline@num}\else 0\fi}%
      \to\labelref@list
```

```
\fi}
```

20 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*

```
\WithSuffix\newcommand\sidenotemargin*[1]{%
  \l@dgetsidenote@margin{#1}
  \global\sidenote@marginR=\@l@dtempcntb
  \global\sidenote@margin=\@l@dtempcntb
}
\newcount\sidenote@marginR
\global\sidenote@margin=\@ne
```

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

```
\newcommand*{\affixside@noteR}{%
    \def\sidenotecontent@{}%
    \numgdef{\itemcount@}{0}%
    \def\do##1{%
        \ifnumequal{\itemcount@}{0}%
            {%
            \appto\sidenotecontent@{##1}}% Not print not separator before the 1st note
            {\appto\sidenotecontent@{\sidenotesep ##1}%
            }%
            \numgdef{\itemcount@}{\itemcount@+1}%
    }%
    \dolistloop{\l@dcsnotetext}%
    \ifnumgreater{\itemcount@}{1}{\led@err@ManySidenotes}{}%
  \gdef\@templ@d{}%
  \gdef\@templ@n{\l@dcsnotetext\l@dcsnotetext@l\l@dcsnotetext@r}%
  \ifx\@templ@d\@templ@n \else%
    \if@twocolumn%
      \if@firstcolumn%
        \setl@dlp@rbox{##1}{\sidenotecontent@}%
      \else%
        \setl@drp@rbox{\sidenotecontent@}%
      \fi%
    \else%
      \@l@dtempcntb=\sidenote@marginR%
      \ifnum\@l@dtempcntb>\@ne%
        \advance\@l@dtempcntb by\page@numR%
      \fi%
      \ifodd\@l@dtempcntb%
        \setl@drp@rbox{\sidenotecontent@}%
```

```
        \gdef\sidenotecontent@{}%
        \numdef{\itemcount@}{0}%
        \dolistloop{\l@dcsnotetext@l}%
        \ifnumgreater{\itemcount@}{1}{\led@err@ManyLeftnotes}{}%
        \setl@dlp@rbox{\sidenotecontent@}%
      \else%
        \setl@dlp@rbox{\sidenotecontent@}%
        \gdef\sidenotecontent@{}%
        \numdef{\itemcount@}{0}%
        \dolistloop{\l@dcsnotetext@r}%
        \ifnumgreater{\itemcount@}{1}{\led@err@ManyRightnotes}{}%
        \setl@drp@rbox{\sidenotecontent@}%
      \fi%
    \fi%
  \fi%
}

```

21 Familiar footnotes

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

```
\renewcommand{\l@dbfnote}[1]{%
  \ifnumberedpar@
  \gdef\@tag{#1}%
    \ifledRcol%
      \xright@appenditem{\noexpand\vl@dbfnote{{\csexpandonce{@tag}}}{\@thefnmark}}%
                        \to\inserts@listR
       \global\advance\insert@countR \@ne%
    \else%
      \xright@appenditem{\noexpand\vl@dbfnote{{\csexpandonce{@tag}}}{\@thefnmark}}%
                        \to\inserts@list
      \global\advance\insert@count \@ne%
    \fi
  \fi\ignorespaces}

```

`\normalbfnoteX`

```
\renewcommand{\normalbfnoteX}[2]{%
  \ifnumberedpar@
    \ifledRcol%
      \ifluatex
          \footnotelang@lua[R]%
      \fi
      \@ifundefined{xpg@main@language}%if polyglossia
        {}%
        {\footnotelang@poly[R]}%
      \protected@csxdef{thisfootnote}{\csuse{thefootnote#1}}%
```

```
      \xright@appenditem{\noexpand\vbfnoteX{#1}{#2}{\csexpandonce{thisfootnote}}}%
                          \to\inserts@listR
       \global\advance\insert@countR \@ne%
    \else%
      \ifluatex
          \footnotelang@lua%
      \fi
      \@ifundefined{xpg@main@language}%if polyglossia
        {}%
        {\footnotelang@poly}%
      \protected@csxdef{thisfootnote}{\csuse{thefootnote#1}}%
      \xright@appenditem{\noexpand\vbfnoteX{#1}{#2}{\csexpandonce{thisfootnote}}}%
                          \to\inserts@list
      \global\advance\insert@count \@ne%
    \fi
  \fi\ignorespaces}
```

22 Verse

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

`\inserthangingsymbolL`
`\inserthangingsymbolR`

```
\newif\ifinserthangingsymbolR
\newcommand{\inserthangingsymbolL}{%
\ifinserthangingsymbol%
    \ifinstanzaL%
        \hangingsymbol%
    \fi%
\fi}
\newcommand{\inserthangingsymbolR}{%
\ifinserthangingsymbolR%
    \ifinstanzaR%
        \hangingsymbol%
    \fi%
\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`

```
\newcommand{\correcthangingL}{%
\ifl@dpaging\else%
    \ifinstanzaL%
        \ifinserthangingsymbol%
            \hskip \@ifundefined{sza@0@}{0}{\expandafter%
```

```
                    \noexpand\csname sza@0@\endcsname}\stanzaindentbase%
        \fi%
    \fi%
\fi}

\newcommand{\correcthangingR}{%
\ifl@dpaging\else%
    \ifinstanzaR%
        \ifinserthangingsymbolR%
            \hskip \@ifundefined{sza@0@}{0}{\expandafter%
                    \noexpand\csname sza@0@\endcsname}\stanzaindentbase%
        \fi%
    \fi%
\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.

```
  \chardef\next=\catcode`\&
  \catcode`\&=\active

```

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

```
\newenvironment{astanza}{%
  \startstanzahook
  \catcode`\&\active
  \global\stanza@count\@ne\stanza@modulo\@ne
  \ifnum\usenamecount{sza@0@}=\z@
    \let\stanza@hang\relax
    \let\endlock\relax
  \else
%%%    \interlinepenalty\@M % this screws things up, but I don't know why
    \rightskip\z@ plus 1fil\relax
  \fi
  \ifnum\usenamecount{szp@0@}=\z@
    \let\sza@penalty\relax
  \fi
  \def&{%
    \endlock\mbox{}%
    \sza@penalty
    \global\advance\stanza@count\@ne
    \@astanza@line}%
  \def\&{%
    \endlock\mbox{}
    \pend
    \endstanzaextra}%
  \pstart
  \@astanza@line
```

```
}{}

```

`\@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.

```
\newcommand*{\@astanza@line}{%
  \ifnum\value{stanzaindentsrepetition}=0
      \parindent=\csname sza@\number\stanza@count
               @\endcsname\stanzaindentbase
  \else
      \parindent=\csname sza@\number\stanza@modulo
               @\endcsname\stanzaindentbase
      \managestanza@modulo
  \fi
  \par
  \stanza@hang%\mbox{}%
  \ignorespaces}

```

Lastly reset the modified category codes.

```
  \catcode`\&=\next

```

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

```
\providecommand*{\newnamebox}[1]{%
  \expandafter\newbox\csname #1\endcsname}
\providecommand*{\setnamebox}[1]{%
  \expandafter\setbox\csname #1\endcsname}
\providecommand*{\unhnamebox}[1]{%
  \expandafter\unhbox\csname #1\endcsname}
\providecommand*{\unvnamebox}[1]{%
  \expandafter\unvbox\csname #1\endcsname}
\providecommand*{\namebox}[1]{%
                      \csname #1\endcsname}

```

`\newnamecount` `\usenamecount` Macros for creating and using 'named' counts.

```
\providecommand*{\newnamecount}[1]{%
  \expandafter\newcount\csname #1\endcsname}
\providecommand*{\usenamecount}[1]{%
                        \csname #1\endcsname}

```

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

```
\newcount\l@dc@maxchunks
\newcommand{\maxchunks}[1]{\l@dc@maxchunks=#1}
  \maxchunks{5120}

```

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

```
\newcount\l@dnumpstartsR

```

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

```
\newcount\l@dpscL
\newcount\l@dpscR

```

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

```
\newcommand*{\l@dsetuprawboxes}{%
  \@l@dtempcntb=\l@dc@maxchunks
  \loop\ifnum\@l@dtempcntb>\z@
    \newnamebox{l@dLcolrawbox\the\@l@dtempcntb}
    \newnamebox{l@dRcolrawbox\the\@l@dtempcntb}
    \advance\@l@dtempcntb \m@ne
  \repeat}

```

`\l@dsetupmaxlinecounts` `\l@dzeromaxlinecounts` 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.

```
\newcommand*{\l@dsetupmaxlinecounts}{%
  \@l@dtempcntb=\l@dc@maxchunks
  \loop\ifnum\@l@dtempcntb>\z@
    \newnamecount{l@dmaxlinesinpar\the\@l@dtempcntb}
    \advance\@l@dtempcntb \m@ne
  \repeat}
\newcommand*{\l@dzeromaxlinecounts}{%
  \begingroup
  \@l@dtempcntb=\l@dc@maxchunks
  \loop\ifnum\@l@dtempcntb>\z@
```

```
    \global\usenamecount{l@dmaxlinesinpar\the\@l@dtempcntb}=\z@
    \advance\@l@dtempcntb \m@ne
  \repeat
  \endgroup}
```

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

```
\AtBeginDocument{%
  \l@dsetuprawboxes
  \l@dsetupmaxlinecounts
  \l@dzeromaxlinecounts
  \l@dnumpstartsL=\z@
  \l@dnumpstartsR=\z@
  \l@dpscL=\z@
  \l@dpscR=\z@}
```

25 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` `\l@dusedbabelfalse` `\l@dusedbabeltrue` A flag for checking if `babel` has been used as a package.

```
\newif\ifl@dusedbabel
  \l@dusedbabelfalse
```

`\ifl@dsamelang` Suppress `\ifl@dsamelang` which didn't work and was not logical, because both columns could have the same language but not the main language of the document.

`\l@dchecklang`

`\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 `\`.

```
\newcommand*{\l@dbbl@set@language}[1]{%
  \edef\languagename{#1}%
  \select@language{\languagename}%
```

```
\if@filesw
  \protected@write\@auxout{}{\string\select@language{\languagename}}%
  \addtocontents{toc}{\string\select@language{\languagename}}%
  \addtocontents{lof}{\string\select@language{\languagename}}%
  \addtocontents{lot}{\string\select@language{\languagename}}%
\fi}
```

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` `\l@duselanguage` `\theledlanguageL` `\theledlanguageR`

`\selectlanguage` is a `babel` command. `\theledlanguageL` and `\theledlanguageR` are the names of the languages of the left and right texts. `\l@duselanguage` is similar to `\selectlanguage`.

```
\providecommand{\selectlanguage}[1]{}
\newcommand*{\l@duselanguage}[1]{}
\gdef\theledlanguageL{}
\gdef\theledlanguageR{}
```

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

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

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

```
   \l@dusedbabelfalse
   \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.

```
   \l@dusedbabeltrue
   \let\l@doldselectlanguage\selectlanguage
   \let\l@doldbbl@set@language\bbl@set@language
   \let\bbl@set@language\l@dbbl@set@language
   \renewcommand{\selectlanguage}[1]{%
     \l@doldselectlanguage{#1}%
     \ifledRcol \gdef\theledlanguageR{#1}%
     \else      \gdef\theledlanguageL{#1}%
     \fi}
```

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

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

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

```
   \gdef\theledlanguageL{\bbl@main@language}%
   \gdef\theledlanguageR{\bbl@main@language}%
```

```
    }%
  }
```

If on Polyglossia

```
 {    \let\old@otherlanguage\otherlanguage%
      \renewcommand{\otherlanguage}[2][]{%
       \selectlanguage[#1]{#2}%
       \ifledRcol  \gdef\theledlanguageR{#2}%
       \else      \gdef\theledlanguageL{#2}%
       \fi}%
     \let\l@duselanguage\xpg@set@language%
     \gdef\theledlanguageL{\xpg@main@language}%
     \gdef\theledlanguageR{\xpg@main@language}%
% \end{macrocode}
% That's it.
%    \begin{macrocode}
}}
```

26 Parallel columns

`\@eledsectionL` `\@eledsectionR` The parbox `\@eledsectionL` and `\@eledsectionR` will keep the sections' title.

```
\newsavebox{\@eledsectionL}%
\newsavebox{\@eledsectionR}%
```

`\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.

```
\newcommand*{\Columns}{%
  \eledsection@correcting@skip=-\baselineskip% Correction for sections' titles
  \setcounter{pstartL}{\value{pstartLold}}
  \setcounter{pstartR}{\value{pstartRold}}
  \ifnum\l@dnumpstartsL=\l@dnumpstartsR\else
    \led@err@BadLeftRightPstarts{\the\l@dnumpstartsL}{\the\l@dnumpstartsR}%
  \fi
```

Start a group and zero counters, etc.

```
  \begingroup
    \l@dzeropenalties
    \endgraf\global\num@lines=\prevgraf
            \global\num@linesR=\prevgraf
    \global\par@line=\z@
    \global\par@lineR=\z@
    \global\l@dpscL=\z@
    \global\l@dpscR=\z@
```

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

```
    \check@pstarts
    \loop\if@pstarts
```

```
1727          \global\pstartnumtrue
1728          \global\pstartnumRtrue
```

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

```
1729        \global\advance\l@dpscL \@ne
1730        \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

```
1731        \checkraw@text
1732 {          \loop\ifaraw@text
```

Grab the next pair of left and right text lines and output them, swapping languages if they differ, adding section title if needed.

```
1733              \l@duselanguage{\theledlanguageL}%
1734              \do@lineL
1735              \xifinlist{\the\l@dpscL}{\eled@sections@@}
1736                  {%
1737                  \ifdefstring{\@eledsectmark}{L}%
1738                    {\csuse{eled@sectmark@\the\l@dpscL}%
1739                    }{}%
1740                  \global\csundef{eled@sectmark@\the\l@dpscL}%
1741                  \savebox{\@eledsectionL}{\parbox[t][][t]{\Lcolwidth}{\vbox{}\print@eledsec
1742                  }%
1743                  {}%
1744              \l@duselanguage{\theledlanguageR}%
1745              \do@lineR
1746              \xifinlist{\the\l@dpscR}{\eled@sectionsR@@}
1747                  {%
1748                  \ifdefstring{\@eledsectmark}{R}%
1749                    {\csuse{eled@sectmark@\the\l@dpscR R}%
1750                    }{}%
1751                  \global\csundef{eled@sectmark@\the\l@dpscR R}%
1752                  \savebox{\@eledsectionR}{\parbox[t][][t]{\Rcolwidth}{\vbox{}\print@eledsec
1753                  {}%
1754            \hb@xt@ \hsize{%
1755             \ifdefstring{\columns@position}{L}{}{\hfill }%
1756              \unhbox\l@dleftbox%
1757              \ifhbox\@eledsectionL%
1758                 \usebox{\@eledsectionL}%
1759              \fi%
1760              \print@columnseparator%
1761              \unhbox\l@drightbox%
1762              \ifhbox\@eledsectionR%
1763                \usebox{\@eledsectionR}%
1764              \fi%
1765              \ifdefstring{\columns@position}{R}{}{\hfill}%
1766            }%
1767            \checkraw@text
1768            \checkverseL
```

```
          \checkverseR
          \checkpb@columns
        \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.

```
      \@writelinesinparL
      \@writelinesinparR
      \check@pstarts
            \ifbypstart@
          \write\linenum@out{\string\@set[1]}
          \resetprevline@
        \fi
       \ifbypstart@R
          \write\linenum@outR{\string\@set[1]}
          \resetprevline@
        \fi
      \addtocounter{pstartL}{1}
      \addtocounter{pstartR}{1}
    \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.

```
    \flush@notes
    \flush@notesR
  \endgroup
  \global\l@dpscL=\z@
  \global\l@dpscR=\z@
  \global\l@dnumpstartsL=\z@
  \global\l@dnumpstartsR=\z@
  \ignorespaces
    \global\instanzaLfalse
    \global\instanzaRfalse}

```

`\print@columnseparator` `\print@columnseparator` prints the column separator, with surrounding spaces (as the user has set them). We use the TEX `\ifdim` instead of etoolbox to avoid having `\hfill` in a `{}`, which deletes some space (but not much).

```
\def\print@columnseparator{%
  \ifdim\beforecolumnseparator<0pt%
    \hfill%
  \else%
    \hspace{\beforecolumnseparator}%
  \fi%
  \columnseparator%
  \ifdim\aftercolumnseparator<0pt%
    \hfill%
  \else%
    \hspace{\beforecolumnseparator}%
```

```
  \fi%
}%
%\end{macrocode}
% \end{macro}
% \begin{macro}{\checkpb@columns}
% \cs{checkpb@columns} prevent or make pagebreaking in columns, depending of the use of \c
%    \begin{macrocode}

\newcommand{\checkpb@columns}{%
    \newif\if@pb
    \newif\if@nopb
    \IfStrEq{\led@pb@setting}{before}{
    \numdef{\next@absline}{\the\absline@num+1}%
    \numdef{\next@abslineR}{\the\absline@numR+1}%
  \xifinlistcs{\next@absline}{l@prev@pb}{\@pbtrue}{}%
  \xifinlistcs{\next@abslineR}{l@prev@pbR}{\@pbtrue}{}
  \xifinlistcs{\next@absline}{l@prev@nopb}{\@nopbtrue}{}%
  \xifinlistcs{\next@abslineR}{l@prev@nopbR}{\@nopbtrue}{}
  }{}
    \IfStrEq{\led@pb@setting}{after}{
  \xifinlistcs{\the\absline@num}{l@prev@pb}{\@pbtrue}{}%
  \xifinlistcs{\the\absline@numR}{l@prev@pbR}{\@pbtrue}{}
  \xifinlistcs{\the\absline@num}{l@prev@nopb}{\@nopbtrue}{}%
  \xifinlistcs{\the\absline@numR}{l@prev@nopbR}{\@nopbtrue}{}
  }{}
\if@nopb\nopagebreak[4]\enlargethispage{\baselineskip}\fi
\if@pb\pagebreak[4]\fi
}
```

`\columnseparator` `\columnrulewidth`

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).

```
\newcommand*{\columnseparator}{%
  \smash{\rule[-0.2\baselineskip]{\columnrulewidth}{1.05\baselineskip}}}
\newdimen\columnrulewidth
  \columnrulewidth=\z@

```

`\columnsposition` `\columns@position`

The position of the `\Columns` in a page. Default value is R. Stored in `\columns@position`.

```
\newcommand*{\columnsposition}[1]{%
  \xdef\columns@position{#1}%
  }%
\xdef\columns@position{R}%
```

`\beforecolumnseparator` `\aftercolumnseparator`

`\beforecolumnseparator` and `\aftercolumnseparator` lengths are defined to `-1pt`. If user changes them to a positive length, the lengths are used to define blank spaces before / after the column separator, instead of `\hfill`.

```
\newlength{\beforecolumnseparator}%
\setlength{\beforecolumnseparator}{-2pt}%

\newlength{\aftercolumnseparator}%
\setlength{\aftercolumnseparator}{-2pt}%
```

`\if@pstarts` `\@pstartstrue` `\@pstartsfalse` `\check@pstarts`

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

```
\newif\if@pstarts
\newcommand*{\check@pstarts}{%
  \@pstartsfalse
  \ifnum\l@dnumpstartsL>\l@dpscL
    \@pstartstrue
  \else
    \ifnum\l@dnumpstartsR>\l@dpscR
      \@pstartstrue
    \fi
  \fi
}

```

`\ifaraw@text` `\araw@texttrue` `\araw@textfalse` `\checkraw@text`

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

```
\newif\ifaraw@text
  \araw@textfalse
\newcommand*{\checkraw@text}{%
  \araw@textfalse
  \ifvbox\namebox{l@dLcolrawbox\the\l@dpscL}
    \araw@texttrue
  \else
    \ifvbox\namebox{l@dRcolrawbox\the\l@dpscR}
      \araw@texttrue
    \fi
  \fi
}

```

`\@writelinesinparL` `\@writelinesinparR`

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

```
\newcommand*{\@writelinesinparL}{%
  \edef\next{%
    \write\linenum@out{\string\@pend[\the\@donereallinesL]}}%
  \next
  \global\@donereallinesL \z@}
\newcommand*{\@writelinesinparR}{%
  \edef\next{%
    \write\linenum@outR{\string\@pendR[\the\@donereallinesR]}}%
  \next
  \global\@donereallinesR \z@}

```

27 Parallel pages

This is considerably more complicated than parallel columns.

\numpagelinesL
\numpagelinesR
\l@dminpagelines

Counts for the number of lines on a left or right page, and the smaller of the number of lines on a pair of facing pages.

```
\newcount\numpagelinesL
\newcount\numpagelinesR
\newcount\l@dminpagelines
```

\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.

```
\newcommand*{\Pages}{%
  \eledsection@correcting@skip=-2\baselineskip% line correcting for section titles.
  \setcounter{pstartL}{\value{pstartLold}}
  \setcounter{pstartR}{\value{pstartRold}}
  \parledgroup@notespacing@set@correction
  \typeout{}
  \typeout{************************** PAGES **************************}
  \ifnum\l@dnumpstartsL=\l@dnumpstartsR\else
    \led@err@BadLeftRightPstarts{\the\l@dnumpstartsL}{\the\l@dnumpstartsR}%
  \fi
```

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

```
  \cleartol@devenpage
  \begingroup
    \l@dzeropenalties
    \endgraf\global\num@lines=\prevgraf
            \global\num@linesR=\prevgraf
    \global\par@line=\z@
    \global\par@lineR=\z@
    \global\l@dpscL=\z@
    \global\l@dpscR=\z@
    \writtenlinesLfalse
    \writtenlinesRfalse
```

Check if there are chunks to be processed.

```
    \check@pstarts
    \loop\if@pstarts
```

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

```
      \global\advance\l@dpscL \@ne
      \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.

```
      \getlinesfromparlistL
```

```
      \getlinesfromparlistR
      \l@dcalc@maxoftwo{\@cs@linesinparL}{\@cs@linesinparR}%
          {\usenamecount{l@dmaxlinesinpar\the\l@dpscL}}%
      \check@pstarts
    \repeat
```

Zero the counts again, ready for the next bit.

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

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

```
    \getlinesfrompagelistL
    \getlinesfrompagelistR
    \l@dcalc@minoftwo{\@cs@linesonpageL}{\@cs@linesonpageR}%
        {\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.

```
    \check@pstarts
    \if@pstarts
```

Increment the chunk counts to get the first pair.

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

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

```
      \global\@donereallinesL=\z@
      \global\@donetotallinesL=\z@
      \global\@donereallinesR=\z@
      \global\@donetotallinesR=\z@
```

Start a loop over the boxes (chunks).

```
      \checkraw@text
%      \begingroup
{        \loop\ifaraw@text
```

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

```
          \checkpageL
          \l@duselanguage{\theledlanguageL}%
%%%          \begingroup
{            \loop\ifl@dsamepage%
```

Process the next (left) text line, adding it to the page. Eventually, adds the optional argument of pstart.

```
              \ifdefstring{\@eledsectnotoc}{L}{\ledsectnotoc}{}
              \csuse{before@pstartL@\the\l@dpscL}
              \global\csundef{before@pstartL@\the\l@dpscL}
              \do@lineL
              \xifinlist{\the\l@dpscL}{\eled@sections@@}
```

```
1947                {\print@eledsectionL}%
1948                {}%
1949              \advance\numpagelinesL \@ne
1950              \ifshiftedpstarts
1951                      \ifdim\ht\l@dleftbox>0pt\hb@xt@ \hsize{\ledstrutL\unhbox\l@dleftbox
1952              \else%
1953                      \parledgroup@correction@notespacing{L}
1954                      \hb@xt@ \hsize{\ledstrutL\unhbox\l@dleftbox}%
1955              \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.

```
1956              \get@nextboxL
1957              \checkpageL
1958              \checkverseL
1959              \checkpbL
1960            \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.

```
1961            \ifl@dpagefull
1962              \@writelinesonpageL{\the\numpagelinesL}%
1963            \else
1964              \@writelinesonpageL{1000}%
1965            \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.

```
1966            \numpagelinesL \z@
1967            \parledgroup@correction@notespacing@init
1968            \clearl@dleftpage }%
```

Now do the same for the right text.

```
1969          \checkpageR
1970          \l@duselanguage{\theledlanguageR}%
1971 {           \loop\ifl@dsamepage%
1972              \initnumbering@sectcountR
1973              \ifdefstring{\@eledsectnotoc}{R}{\ledsectnotoc}{}
1974              \csuse{before@pstartR@\the\l@dpscR}
1975              \global\csundef{before@pstartR@\the\l@dpscR}
1976              \do@lineR
1977              \xifinlist{\the\l@dpscR}{\eled@sectionsR@@}
1978                {\print@eledsectionR}%
1979                {}%
1980              \advance\numpagelinesR \@ne
1981              \ifshiftedpstarts
1982                      \ifdim\ht\l@drightbox>0pt\hb@xt@ \hsize{\ledstrutR\unhbox\l@drightbo
1983              \else%
```

```
                    \parledgroup@correction@notespacing{R}
                    \hb@xt@ \hsize{\ledstrutR\unhbox\l@drightbox}%
              \fi
              \get@nextboxR
              \checkpageR
              \checkverseR
              \checkpbR
            \repeat
            \ifl@dpagefull
              \@writelinesonpageR{\the\numpagelinesR}%
            \else
              \@writelinesonpageR{1000}%
            \fi
            \numpagelinesR=\z@
            \parledgroup@correction@notespacing@init
```

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

```
            \clearl@drightpage}
```

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

```
          \checkraw@text
          \ifaraw@text
            \getlinesfrompagelistL
            \getlinesfrompagelistR
            \l@dcalc@minoftwo{\@cs@linesonpageL}{\@cs@linesonpageR}%
                             {\l@dminpagelines}%
          \fi
        \repeat}
```

We have now output the text from all the chunks.

```
    \fi
```

Make sure that there are no inserts hanging around.

```
    \flush@notes
    \flush@notesR
  \endgroup
```

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

```
  \global\l@dpscL=\z@
  \global\l@dpscR=\z@
  \global\l@dnumpstartsL=\z@
  \global\l@dnumpstartsR=\z@
    \global\instanzaLfalse
    \global\instanzaRfalse
  \ignorespaces}
```

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

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

\cleartoevenpage
\cleartol@devenpage
\clearl@dleftpage
\clearl@drightpage

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

```
\providecommand{\cleartoevenpage}[1][\@empty]{%
  \clearpage
  \ifodd\c@page\hbox{}#1\clearpage\fi}
\newcommand*{\cleartol@devenpage}{%
  \ifdim\pagetotal<\topskip%  on an empty page
  \else
    \clearpage
  \fi
  \ifodd\c@page\hbox{}\clearpage\fi}
\newcommand*{\clearl@dleftpage}{%
  \clearpage
  \ifodd\c@page\else
    \led@err@LeftOnRightPage
    \hbox{}%
    \cleardoublepage
  \fi}
\newcommand*{\clearl@drightpage}{%
  \clearpage
  \ifodd\c@page
    \led@err@RightOnLeftPage
    \hbox{}%
    \cleartoevenpage
  \fi}
```

\getlinesfromparlistL
\@cs@linesinparL
\getlinesfromparlistR
\@cs@linesinparR

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

```
\newcommand*{\getlinesfromparlistL}{%
  \ifx\linesinpar@listL\empty
    \gdef\@cs@linesinparL{0}%
  \else
    \gl@p\linesinpar@listL\to\@cs@linesinparL
  \fi}
\newcommand*{\getlinesfromparlistR}{%
  \ifx\linesinpar@listR\empty
    \gdef\@cs@linesinparR{0}%
  \else
    \gl@p\linesinpar@listR\to\@cs@linesinparR
  \fi}
```

```
2059
```

\getlinesfrompagelistL
\@cs@linesonpageL
\getlinesfrompagelistR
\@cs@linesonpageR

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

```
2060 \newcommand*{\getlinesfrompagelistL}{%
2061   \ifx\linesonpage@listL\empty
2062     \gdef\@cs@linesonpageL{1000}%
2063   \else
2064     \gl@p\linesonpage@listL\to\@cs@linesonpageL
2065   \fi}
2066 \newcommand*{\getlinesfrompagelistR}{%
2067   \ifx\linesonpage@listR\empty
2068     \gdef\@cs@linesonpageR{1000}%
2069   \else
2070     \gl@p\linesonpage@listR\to\@cs@linesonpageR
2071   \fi}
2072
```

\@writelinesonpageL
\@writelinesonpageR

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

```
2073 \newcommand*{\@writelinesonpageL}[1]{%
2074   \edef\next{\write\linenum@out{\string\@lopL{#1}}}%
2075   \next}
2076 \newcommand*{\@writelinesonpageR}[1]{%
2077   \edef\next{\write\linenum@outR{\string\@lopR{#1}}}%
2078   \next}
2079
```

\l@dcalc@maxoftwo
\l@dcalc@minoftwo

\l@dcalc@maxoftwo{⟨*num*⟩}{⟨*num*⟩}{⟨*count*⟩} sets ⟨*count*⟩ to the maximum of the two ⟨*num*⟩.

Similarly \l@dcalc@minoftwo{⟨*num*⟩}{⟨*num*⟩}{⟨*count*⟩} sets ⟨*count*⟩ to the minimum of the two ⟨*num*⟩.

```
2080 \newcommand*{\l@dcalc@maxoftwo}[3]{%
2081   \ifnum #2>#1\relax
2082     #3=#2\relax
2083   \else
2084     #3=#1\relax
2085   \fi}
2086 \newcommand*{\l@dcalc@minoftwo}[3]{%
2087   \ifnum #2<#1\relax
2088     #3=#2\relax
2089   \else
2090     #3=#1\relax
2091   \fi}
2092
```

\ifl@dsamepage
\l@dsamepagetrue
\l@dsamepagefalse
\ifl@dpagefull
\l@dpagefulltrue
\l@dpagefullfalse
\checkpageL
\checkpageR

\checkpageL tests if the space and lines already taken on the page by text and footnotes is less than the constraints. If so, then \ifl@dpagefull is set FALSE and

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

```
\newif\ifl@dsamepage
  \l@dsamepagetrue
\newif\ifl@dpagefull

\newcommand*{\checkpageL}{%
  \l@dpagefulltrue
  \l@dsamepagetrue
  \check@goal
  \ifdim\pagetotal<\ledthegoal
    \ifnum\numpagelinesL<\l@dminpagelines
    \else
      \l@dsamepagefalse
      \l@dpagefullfalse
    \fi
  \else
    \l@dsamepagefalse
    \l@dpagefulltrue
  \fi}
\newcommand*{\checkpageR}{%
  \l@dpagefulltrue
  \l@dsamepagetrue
  \check@goal
  \ifdim\pagetotal<\ledthegoal
    \ifnum\numpagelinesR<\l@dminpagelines
    \else
      \l@dsamepagefalse
      \l@dpagefullfalse
    \fi
  \else
    \l@dsamepagefalse
    \l@dpagefulltrue
  \fi}

```

\checkpbL
\checkpbR

\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.

```
\newcommand{\checkpbL}{
   \IfStrEq{\led@pb@setting}{after}{
     \xifinlistcs{\the\absline@num}{l@prev@pb}{\l@dpagefulltrue\l@dsamepagefalse}{}
     \xifinlistcs{\the\absline@num}{l@prev@nopb}{\l@dpagefullfalse\l@dsamepagetrue}{}
   }{}
   \IfStrEq{\led@pb@setting}{before}{
     \numdef{\next@absline}{\the\absline@num+1}
     \xifinlistcs{\next@absline}{l@prev@pb}{\l@dpagefulltrue\l@dsamepagefalse}{}
```

```
      \xifinlistcs{\next@absline}{l@prev@nopb}{\l@dpagefullfalse\l@dsamepagetrue}{}
    }{}
}

\newcommand{\checkpbR}{
    \IfStrEq{\led@pb@setting}{after}{
      \xifinlistcs{\the\absline@numR}{l@prev@pbR}{\l@dpagefulltrue\l@dsamepagefalse}{}
      \xifinlistcs{\the\absline@numR}{l@prev@nopbR}{\l@dpagefullfalse\l@dsamepagetrue}{}
      }{}
    \IfStrEq{\led@pb@setting}{before}{
      \numdef{\next@abslineR}{\the\absline@numR+1}
      \xifinlistcs{\next@abslineR}{l@prev@pbR}{\l@dpagefulltrue\l@dsamepagefalse}{}
      \xifinlistcs{\next@abslineR}{l@prev@nopbR}{\l@dpagefullfalse\l@dsamepagetrue}{}
    }{}
}
```

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

```
\newcommand{\checkverseL}{
\ifinstanzaL
  \ifledpbinverse
    \ifinserthangingsymbol
      \numgdef{\prev@abslineverse}{\the\absline@num-1}
      \IfStrEq{\led@pb@setting}{after}{\lednopbnum{\prev@abslineverse}}{}
      \IfStrEq{\led@pb@setting}{before}{\ifnum\numpagelinesL<3\ledpbnum{\prev@abslineverse}\fi}{}
    \fi
  \fi
\fi
}
\newcommand{\checkverseR}{
\ifinstanzaR
  \ifledpbinverse
    \ifinserthangingsymbolR
      \numgdef{\prev@abslineverse}{\the\absline@numR-1}
      \IfStrEq{\led@pb@setting}{after}{\lednopbnumR{\prev@abslineverse}}{}
      \IfStrEq{\led@pb@setting}{before}{\ifnum\numpagelinesR<3\ledpbnumR{\prev@abslineverse}\fi}{}
    \fi
  \fi
\fi
}
```

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

```
\newdimen\ledthegoal
\ifshiftedpstarts
       \newcommand*{\goalfraction}{0.95}
\else
       \newcommand*{\goalfraction}{0.9}
```

```
\fi

\newcommand*{\check@goal}{%
  \ledthegoal=\goalfraction\pagegoal}

```

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

\ifwrittenlinesL
```
\newif\ifwrittenlinesL
\newif\ifwrittenlinesR

```

\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.

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

The current box is not empty; do nothing.

```
  \else%                                        box is empty
```

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

```
    \ifnum\usenamecount{l@dmaxlinesinpar\the\l@dpscL}>\@donetotallinesL
      \parledgroup@notes@endL
    \else
```

Sufficient lines have been output.

```
      \ifnum\usenamecount{l@dmaxlinesinpar\the\l@dpscL}=\@donetotallinesL
        \parledgroup@notes@endL
      \fi
      \ifwrittenlinesL\else
```

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

```
        \@writelinesinparL
        \writtenlinesLtrue
      \fi
      \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.

```
        \writtenlinesLfalse
         \ifbypstart@
          \ifnum\value{pstartL}<\value{pstartLold}
          \else
            \global\line@num=0
            \resetprevline@
          \fi
          \fi
% Add the content of the optional argument of the previous \cs{pend}.
%     \begin{macrocode}
          \csuse{after@pendL@\the\l@dpscL}%
          \global\csundef{after@pendL@\the\l@dpscL}%
```

```
% \end{macrocode}
% \begin{macrocode}
  \addtocounter{pstartL}{1}
  \global\pstartnumtrue
  \l@dcalc@maxoftwo{\the\usenamecount{l@dmaxlinesinpar\the\l@dpscL}}%
                   {\the\@donetotallinesL}%
                   {\usenamecount{l@dmaxlinesinpar\the\l@dpscL}}%
  \global\@donetotallinesL \z@
  \global\advance\l@dpscL \@ne
```

Add notes of parallel ledgroup.

```
  \parledgroup@notes@endL
  \parledgroup@correction@notespacing@final{L}
 \else
% Add the content of the optional argument of the last \cs{pend}.
% \begin{macrocode}
    \l@dpagefulltrue
    \csuse{after@pendL@\the\l@dpscL}%
    \global\csundef{after@pendL@\the\l@dpscL}%
% \end{macrocode}
% \begin{macrocode}
 \fi
 \fi
\fi}
\newcommand*{\get@nextboxR}{%
 \ifvbox\namebox{l@dRcolrawbox\the\l@dpscR}% box is not empty
 \else%                                      box is empty
  \ifnum\usenamecount{l@dmaxlinesinpar\the\l@dpscR}>\@donetotallinesR
   \parledgroup@notes@endR
  \else
   \ifnum\usenamecount{l@dmaxlinesinpar\the\l@dpscR}=\@donetotallinesR
    \parledgroup@notes@endR
   \fi
   \ifwrittenlinesR\else
    \@writelinesinparR
    \writtenlinesRtrue
   \fi
   \ifnum\l@dnumpstartsR>\l@dpscR
    \writtenlinesRfalse
    \ifbypstart@R
        \ifnum\value{pstartR}<\value{pstartRold}
        \else
           \global\line@numR=0
           \resetprevline@
        \fi
    \fi
    \csuse{after@pendR@\the\l@dpscR}%
    \global\csundef{after@pendR@\the\l@dpscR}%
    \addtocounter{pstartR}{1}
    \global\pstartnumRtrue
```

```
      \l@dcalc@maxoftwo{\the\usenamecount{l@dmaxlinesinpar\the\l@dpscR}}%
                       {\the\@donetotallinesR}%
                       {\usenamecount{l@dmaxlinesinpar\the\l@dpscR}}%
      \global\@donetotallinesR \z@
      \global\advance\l@dpscR \@ne
      \parledgroup@notes@endR
      \parledgroup@correction@notespacing@final{R}
    \else
      \csuse{after@pendR@\the\l@dpscR}%
      \global\csundef{after@pendR@\the\l@dpscR}%
      \l@dpagefulltrue
    \fi
  \fi
\fi}
```

28 Sections' titles' commands

\eledsectnotoc `\eledsectnotoc` just saves its content `\@eledsectnotoc`, which will be tested where sectioning commands will be printed.

```
\newcommand{\eledsectnotoc}[1]{\xdef\@eledsectnotoc{#1}}
\eledsectnotoc{R}
```

\eledsectmark `\eledsectmark` just saves its content `\@eledsectmark`, which will be tested where sectioning commands will be printed.

```
\newcommand{\eledsectmark}[1]{\xdef\@eledsectmark{#1}}
\eledsectmark{L}
```

\eledsection@correcting@skip Because the vertical correction needed after inserting a title in parallel depends whether we are in parallel columns or parallel pages, we stock its length in `\eledsection@correcting@skip`.

```
\newskip\eledsection@correcting@skip
```

We save the sectioning commands of the right side in the `\eled@sectioningR@out` file.

```
\newwrite\eled@sectioningR@out
```

29 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 \prev@nopbR 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).

```
\def\l@prev@pbR{}
\def\l@prev@nopbR{}
```

\ledpbR
\ledpbnumR
\lednopbnum
\lednopbnumR

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.

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

\led@pbR
\led@pbnumR
\led@nopbR
\led@nopbnumR

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.

```
\newcommand{\led@pbR}{\listcsxadd{l@prev@pbR}{\the\absline@numR}}
\newcommand{\led@pbnumR}[1]{\listcsxadd{l@prev@pbR}{#1}}
\newcommand{\led@nopbR}{\listcsxadd{l@prev@nopbR}{\the\absline@numR}}
\newcommand{\led@nopbnumR}[1]{\listcsxadd{l@prev@nopbR}{#1}}
```

30 Parallel ledgroup

\parledgroup@
\parledgroupseries@
\parledgrouptype@

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

```
\newmarks\parledgroup@
\newmarks\parledgroup@series
\newmarks\parledgroup@type
```

\parledgroup@notes@startL
\parledgroup@notes@startR

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

```
\newcommand{\parledgroup@notes@startL}{%
  \ifnum\usenamecount{l@dmaxlinesinpar\the\l@dpscL}>0%
    \IfStrEq{\splitfirstmarks\parledgroup@type}{footnoteX}{\csuse{bhooknoteX@\splitfirstmarks\parledg
    \IfStrEq{\splitfirstmarks\parledgroup@type}{Xfootnote}{\csuse{bhookXnote@\splitfirstmarks\parledg
  \fi%
  \global\ledgroupnotesL@true%
  \insert@noterule@ledgroup{L}%
}
\newcommand{\parledgroup@notes@startR}{%
  \ifnum\usenamecount{l@dmaxlinesinpar\the\l@dpscR}>0%
    \IfStrEq{\splitfirstmarks\parledgroup@type}{footnoteX}{\csuse{bhooknoteX@\splitfirstmarks\parledg
    \IfStrEq{\splitfirstmarks\parledgroup@type}{Xfootnote}{\csuse{bhookXnote@\splitfirstmarks\parledg
  \fi%
  \global\ledgroupnotesR@true%
  \insert@noterule@ledgroup{R}%
}
```

\parledgroup@notes@startL
\parledgroup@notes@startR

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

```
\newcommand{\parledgroup@notes@endL}{%
  \global\ledgroupnotesL@false%
}
\newcommand{\parledgroup@notes@endR}{%
  \global\ledgroupnotesR@false%
}
```

\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

```
\newcommand{\insert@noterule@ledgroup}[1]{
    \IfStrEq{\splitbotmarks\parledgroup@}{begin}{%
      \IfStrEq{\splitbotmarks\parledgroup@type}{Xfootnote}{
        \csuse{ifledgroupnotes#1@}
          \vskip\skip\csuse{mp\splitbotmarks\parledgroup@series footins}
          \csuse{\splitbotmarks\parledgroup@series footnoterule}
        \fi
        }
        {}
        \IfStrEq{\splitbotmarks\parledgroup@type}{footnoteX}{
          \csuse{ifledgroupnotes#1@}
           \vskip\skip\csuse{mpfootins\splitbotmarks\parledgroup@series}
           \csuse{footnoterule\splitbotmarks\parledgroup@series}
           \fi
           }{}
        }
    {}
}
```

\parledgroupnotespacing

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

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

\parledgroup@notespacing@correction
\arledgroup@notespacing@set@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 begining of \Pages.

```
\dimdef{\parledgroup@notespacing@correction}{0pt}
\newcommand{\parledgroup@notespacing@set@correction}{%
  {\notefontsetup\parledgroupnotespacing\dimgdef{\temp@spacing}{\baselineskip}}%
  \dimgdef{\parledgroup@notespacing@correction}{\baselineskip-\temp@spacing}%
}
```

\ledgroup@correction@notespacing@init

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

```
\newcommand{\parledgroup@correction@notespacing@init}{
```

```
  \dimdef{\parledgroup@notespacing@correction@accumulated}{0pt}
  \dimdef{\parledgroup@notespacing@correction@modulo}{0pt}
}
\parledgroup@correction@notespacing@init
```

rection@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.

```
\newcommand{\parledgroup@correction@notespacing@final}[1]{
    \ifparledgroup
    \vspace{\parledgroup@notespacing@correction@accumulated}
    \parledgroup@correction@notespacing@init%
    \ifstrequal{#1}{L}{
        \numdef{\@checking}{\the\l@dpscL-1}
    }{
      \numdef{\@checking}{\the\l@dpscR-1}
    }
    \dimdef{\@beforenotes@current@diff}{\csuse{@parledgroup@beforenotes@\@checking L}-\csuse{@parledg
    \ifstrequal{#1}{L}%
      {% Left
      \ifdimgreater{\@beforenotes@current@diff}{0pt}{}{\vspace{-\@beforenotes@current@diff}}%
      }%
      {% Right
      \ifdimgreater{\@beforenotes@current@diff}{0pt}{\vspace{\@beforenotes@current@diff}}{}
      }%
  \fi
}
```

up@correction@notespacing

`\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.

```
{}
\newcommand{\parledgroup@correction@notespacing}[1]{%
    \csuse{ifledgroupnotes#1@}%
      \vspace{-\parledgroup@notespacing@correction}%
      \dimdef{\parledgroup@notespacing@correction@accumulated}{\parledgroup@notespacing@correction@ac
      \dimdef{\parledgroup@notespacing@correction@modulo}{\parledgroup@notespacing@correction@modulo+
```

```
      \ifdimless{\parledgroup@notespacing@correction@modulo}{\baselineskip}{}{\advance\nu
      \dimdef{\parledgroup@notespacing@correction@modulo}{\parledgroup@notespacing@correct
     }% mean greater than equal
    \fi%
}
```

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

```
\dimdef\parledgroup@beforenotesL{0pt}
\dimdef\parledgroup@beforenotesR{0pt}
```

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

```
\newcommand{\parledgroup@beforenotes@save}[1]{
  \ifparledgroup
    \csdimgdef{@parledgroup@beforenotes@\the\csuse{l@dnumpstarts#1}#1}{\csuse{parledgroup@
    \csdimgdef{parledgroup@beforenotes#1}{0pt}
  \fi
}
```

31 The End

¡/code¿

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

`\&` 1556, 1557, 1561, 1578, 1599

`\@M` 1567

`\@adv` 361, 626, 627

`\@afterindentfalse` 751

`\@arabic` 217, 218, 800, 803

\@astanza@line 1577, 1583, 1586
\@auxout 1390, 1402, 1664
\@beforenotes@current@diff 2352, 2355, 2358
\@chapter 752
\@checking 2348, 2350, 2352
\@cs@linesinparL 1917, 2047
\@cs@linesinparR 1917, 2047
\@cs@linesonpageL . . 1925, 2004, 2060
\@cs@linesonpageR . . 1925, 2004, 2060
\@currentlabel 843, 886
\@donereallinesL 943, 972, 1877, 1879, 1931
\@donereallinesR 943, 1038, 1882, 1884, 1933
\@donetotallinesL 943, 973, 976, 1932, 2187, 2190, 2215, 2217
\@donetotallinesR 943, 1039, 1042, 1934, 2235, 2238, 2259, 2261
\@eled@sectioningfalse 996
\@eled@sectioningtrue 994
\@eledsectionL . 1708, 1741, 1757, 1758
\@eledsectionR . 1708, 1752, 1762, 1763
\@eledsectmark . 991, 1737, 1748, 2275
\@eledsectnotoc 990, 1942, 1973, 2273
\@insertR 1344–1346, 1359–1361
\@l@dtempcnta 434, 436, 438, 439, 443, 445, 447, 448, 1096, 1135, 1136, 1138, 1140, 1143, 1144, 1161–1165, 1167, 1174, 1179, 1183, 1191, 1196, 1200, 1233, 1236, 1238, 1242
\@l@dtempcntb 173, 175, 177, 1126, 1127, 1174, 1179, 1183, 1191, 1196, 1200, 1225, 1229, 1242, 1250–1252, 1254, 1275–1277, 1279, 1296–1298, 1300, 1431, 1432, 1460–1462, 1464, 1627–1631, 1635–1638, 1642–1645
\@lab 553, 1381, 1393, 1418
\@lock 960, 1076
\@lockR . 56, 305, 307, 309, 322, 468, 484, 485, 487, 488, 516, 517, 519, 1025, 1058, 1102, 1104, 1105, 1107, 1188, 1205, 1207, 1209
\@lopL 577, 2074
\@lopR 577, 2077
\@nl 283, 609, 611
\@nl@reg 332
\@nl@regR 283
\@nobreakfalse 809, 853, 993
\@nobreaktrue . 807, 811, 851, 855, 993
\@nopbtrue 1824, 1825, 1830, 1831
\@oldnobreak 807, 809, 851, 853, 904, 925
\@pbtrue 1822, 1823, 1828, 1829
\@pend 568, 1877
\@pendR 568, 1882
\@pstartsfalse 1850
\@pstartstrue 1850
\@ref 540
\@ref@reg 566
\@schapter 752
\@set 393, 633, 634, 1776, 1780
\@startstanza 764, 765, 787, 788
\@tag 663, 686, 1485
\@templ@d 1450, 1452
\@templ@n 1451, 1452
\@writelinesinparL . . 1772, 1875, 2194
\@writelinesinparR . . 1773, 1875, 2242
\@writelinesonpageL . 1962, 1964, 2073
\@writelinesonpageR . 1993, 1995, 2073
\@xloop 1357

A

\absline@num 357, 427, 441, 460, 1067, 1820, 1828, 1830, 2128, 2129, 2132, 2153
\absline@numR 51, 234, 285, 288, 291, 424, 432, 453, 472, 506, 534, 545, 1049, 1088, 1089, 1126, 1343, 1821, 1829, 1831, 2140, 2141, 2144, 2164, 2285, 2287
\actionlines@list 275, 278, 427, 441, 460
\actionlines@listR 238, 253, 267, 270, 424, 432, 453, 472, 506, 534, 1148, 1151
\actions@list . 279, 428, 448, 462, 464
\actions@listR 238, 254, 271, 425, 439, 455, 457, 474, 483, 508, 515, 535, 1152
\add@inserts 983, 988
\add@inserts@nextR 1332
\add@insertsR 1332
\add@penaltiesL 971, 1353
\add@penaltiesR 1037, 1353
\addtocontents 1665–1667
\addtocounter 906, 927, 989, 1783, 1784, 2212, 2256
\advancelabel@refs 1388, 1400

\advanceline 625, 656
\affixline@num 967
\affixline@numR 1032, 1158
\affixpstart@numL 981, 1265
\affixpstart@numR 1265
\affixside@note 983, 988
\affixside@noteR 1437
\aftercolumnseparator . *4*, 1804, 1845
\appto 1443, 1444
\araw@textfalse 1862
\araw@texttrue 1862
astanza (environment) *8*, 1559
\AtBeginDocument . . . 1414, 1649, 1675

B

\ballast@count 1086, 1091
\bbl@main@language 1691, 1692
\bbl@set@language 1682, 1683
\beforecolumnseparator *4*, 1798, 1801, 1807, 1845
\beginnumbering *7*, 775, 814
\beginnumberingR . . . 42, 118, 775, 858
\bfseries 800, 803
\bypage@Rfalse 138, 153, 158
\bypage@Rtrue 138, 148
\bypstart@Rfalse 138, 149, 159
\bypstart@Rtrue 138, 154

C

\c@ballast 1091
\c@chapter 100
\c@chapterR 100
\c@firstlinenumR 182, 1231
\c@firstsublinenumR 186, 1226
\c@linenumincrementR 182, 1231
\c@page 609, 611, 2025, 2031, 2034, 2041
\c@pstart 1403
\c@pstartL 800
\c@pstartR 803, 1391
\c@section 101
\c@sectionR 101
\c@sublinenumincrementR . . 186, 1226
\c@subsection 102
\c@subsectionR 102
\c@subsubsection 103
\c@subsubsectionR 103
\ch@ck@l@ckR 1158
\ch@cksub@l@ckR 1158
\ch@cksub@lockR 1227
\chapter 737, 738, 747
\chapterinpages 729, 738, 749
\chardef 1556
\check@goal 2100, 2114, 2171
\check@pstarts 1725, 1774, 1850, 1911, 1919, 1927
\checkpageL 1938, 1957, 2093
\checkpageR 1969, 1988, 2093
\checkpb@columns . . . 1770, 1812, 1816
\checkpbL 1959, 2126
\checkpbR 1990, 2126
\checkraw@text 1731, 1767, 1862, 1935, 2000
\checkverseL 1768, 1958, 2149
\checkverseR 1769, 1989, 2149
\cleardoublepage 2037
\clearl@dleftpage 1968, 2023
\clearl@drightpage 1999, 2023
\cleartoevenpage 2023
\cleartol@devenpage 1900, 2023
\closeout 92, 590, 596, 599, 603
\columnrulewidth *4*, 1836
\Columns *3*, 1710
\columns@position . . 1755, 1765, 1841
\columnseparator *4*, 1803, 1836
\columnsposition *4*, 1841
\correcthangingL 985, 1537
\correcthangingR 1537
\countLline 938, 949
\countRline 938, 1014
\critext 661
\cs 1813, 2206, 2222
\csdimgdef 2377, 2378
\csexpandonce . . 1487, 1491, 1507, 1518
\csgdef 847, 890, 911, 932
\csundef 997, 1740, 1751, 1944, 1975, 2209, 2226, 2255, 2267
\csuse 995, 1506, 1517, 1738, 1749, 1943, 1974, 2208, 2225, 2254, 2266, 2294, 2295, 2302, 2303, 2317–2319, 2324–2326, 2352, 2364, 2377

D

\DeclareOption 8–10
\def@tempb 156
\dimdef 2333, 2339, 2340, 2352, 2366, 2367, 2369, 2373, 2374
\dimen 612, 613, 617–619, 623
\dimgdef 2335, 2336
\divide 1163

\do@actions 1068
\do@actions@fixedcodeR 1095
\do@actions@nextR 1095
\do@actionsR 1050, 1095
\do@ballast 1069
\do@ballastR 1051, 1086
\do@insidelineLhook 985, 1007
\do@insidelineRhook 1007
\do@lineL 948, 1734, 1945
\do@lineLhook 953, 1007
\do@lineR 1012, 1745, 1976
\do@lineRhook 1007, 1018
\do@lockoff 503
\do@lockoffL 527
\do@lockoffR 503
\do@lockon 468
\do@lockonL 500
\do@lockonR 468
\dolistloop 1448, 1468, 1475
\dummy@ref 549

E

\edfont@info 712, 715, 721, 724
\edlabel 1379
\edtext 684
\eled@sectioningR@out . . 69, 92, 2278
\eled@sections@@ 968, 993, 1735, 1946
\eled@sectionsR@@ 67, 1033, 1746, 1977
\eledpar@error 21, 30, 33, 35
\eledsection@correcting@skip 1005, 1711, 1891, 2277
\eledsectmark *13*, 2275
\eledsectnotoc *13*, 2273
\empty . . 78, 81, 267, 275, 674, 697, 710, 719, 823, 867, 1148, 1230, 1238, 1334–1336, 1347, 1358, 1382, 1394, 2048, 2054, 2061, 2067
\end@lemmas 674, 675, 697, 698
\endashchar 1369
\endgraf 900, 921, 1719, 1903
\endline@num 556, 562
\endlock 645, 1565, 1574, 1579
\endnumbering *7*, 71, 122, 776
\endnumberingR 45, 71, 106, 117, 130, 776
\endpage@num 555, 562
\endstanzaextra 1581
\endsub 612
\endsubline@num 557, 563
\enlargethispage 1833
environments:
astanza *8*, 1559
Leftside *5*, 756
pages *4*, 729
pairs *3*, 729
Rightside *5*, 773
\extensionchars 62, 112, 127, 135

F

\f@x@l@cksR 1158
\first@linenum@out@Rfalse . . 585, 591
\first@linenum@out@Rtrue 585
\firstlinenum *5*, 191
\firstlinenum* *5*, 191
\firstsublinenum *5*, 191
\firstsublinenum* *5*, 191
\fix@page 328, 335
\flag@end 612, 670, 680, 681, 693, 703, 704
\flag@start 612, 669, 670, 681, 692, 693, 704
\flush@notes 1786, 2009
\flush@notesR 1356, 1787, 2010
\footnotelang@lua 1501, 1512
\footnotelang@poly 1505, 1516
\fullstop . 230, 1366, 1368, 1370, 1372

G

\get@linelistfile 263
\get@nextboxL 1956, 2184
\get@nextboxR 1987, 2184
\getline@numL 959, 1066
\getline@numR 1024, 1048
\getlinesfrompagelistL 1923, 2002, 2060
\getlinesfrompagelistR 1924, 2003, 2060
\getlinesfromparlistL . . . 1915, 2047
\getlinesfromparlistR . . . 1916, 2047
\gl@p 270, 271, 278, 279, 675, 698, 714, 723, 1151, 1152, 1340, 1344, 1359, 1385, 1397, 2051, 2057, 2064, 2070
\goalfraction *5*, 2171

H

\hangingsymbol *10*, 1528, 1534
\hb@xt@ 966, 975, 985, 1031, 1041, 1754, 1951, 1954, 1982, 1985

\hsize 841, 884, 1754, 1951, 1954, 1982, 1985

I

\if@filesw 1663
\if@firstcolumn 1244, 1269, 1290, 1454
\if@ledgroup 595
\if@nobreak 806, 850
\if@nopb 1818, 1833
\if@pb 1817, 1834
\if@pstarts 1726, 1850, 1912, 1928
\if@RTL 670, 681, 693, 704, 998
\ifaraw@text . . . 1732, 1862, 1937, 2001
\ifautopar 833, 877
\ifbypage@ 351
\ifbypage@R 138, 341, 1130
\ifbypstart@ 570, 1775, 2199
\ifbypstart@R . . . 138, 574, 1779, 2247
\ifdefstring 990, 991, 1737, 1748, 1755, 1765, 1942, 1973
\ifdim . . . 613, 617, 619, 623, 1798, 1804, 1951, 1982, 2027, 2101, 2115
\ifdimgreater 2355, 2358
\ifdimless 2368
\iffirst@linenum@out@R 585, 589
\ifhbox 1757, 1762
\ifinserthangingsymbol 1526, 1540, 2152
\ifinserthangingsymbolR 1524, 1532, 1550, 2163
\ifinstanzaL 754, 754, 1527, 1539, 2150
\ifinstanzaR 754, 755, 1533, 1549, 2161
\ifl@d@dash 1369
\ifl@d@elin 1371, 1372
\ifl@d@esl 1372
\ifl@d@pnum 1366, 1370
\ifl@d@ssub 1368
\ifl@dpagefull 1961, 1992, 2093
\ifl@dpaging 12, 1538, 1548
\ifl@dpairing 12, 75
\ifl@dsamelang 1660
\ifl@dsamepage 1941, 1971, 2093
\ifl@dskipnumber 1221
\ifl@dusedbabel 1658
\iflabelpstart 843, 886
\ifledgroupnotesL@ 1070
\ifledgroupnotesR@ 1052, 1220
\ifllednopbinverse 2151, 2162
\ifledplinenum 1367
\ifledRcol 12, 174, 196, 200, 204, 208, 250, 265, 329, 338, 363, 377, 394, 411, 423, 431, 452, 497, 524, 533, 542, 614, 620, 626, 633, 641, 646, 650, 655, 665, 688, 709, 1380, 1419, 1486, 1499, 1686, 1698
\ifluatex 1500, 1511
\ifnoteschanged@ 85
\ifnumberedpar@ 816, 860, 896, 917, 1484, 1498
\ifnumbering 143, 812, 893
\ifnumberingR . . . 43, 72, 108, 856, 914
\ifnumberline 1052, 1070, 1220
\ifnumberpstart . . . 834, 878, 905, 926
\ifnumequal 1441
\ifnumgreater 1449, 1469, 1476
\ifodd 1254, 1279, 1300, 1464, 2025, 2031, 2034, 2041
\ifparledgroup 2344, 2376
\ifpst@rtedL 38, 820
\ifpst@rtedR 38, 864
\ifpstartnum 1310, 1315
\ifpstartnumR 1265
\ifshiftedpstarts 5, 1950, 1981, 2172
\ifsidepstartnum 835, 879, 1267, 1288
\ifstrempty 845, 888, 909, 930
\IfStrEq 957, 1022, 1819, 1827, 2127, 2131, 2139, 2143, 2154, 2155, 2165, 2166, 2294, 2295, 2302, 2303, 2315, 2316, 2323
\ifstrequal 2347, 2353
\ifsublines@ 228, 317, 362, 395, 402, 433, 442, 454, 461, 473, 507, 561, 563, 1053, 1071, 1137, 1224, 1421, 1425
\ifvbox 950, 1015, 1866, 1869, 2185, 2233
\ifvmode 1387, 1399
\ifwrittenlinesL 2181, 2193
\ifwrittenlinesR 2182, 2241
\initnumbering@sectcmd 114, 732, 741
\initnumbering@sectcountR 66, 95, 1972
\InputIfFileExists 68
\insert@count 539, 666, 689, 1493, 1520
\insert@countR 540, 665, 688, 1489, 1509
\insert@noterule@ledgroup 2298, 2306, 2314
\inserthangingsymbolfalse 963
\inserthangingsymbolL 985, 1524
\inserthangingsymbolR 1524

\inserthangingsymbolRfalse 1028
\inserthangingsymbolRtrue 1026
\inserthangingsymboltrue 961
\insertlines@listR .....
.... 78, 238, 252, 545, 1336, 1340
\inserts@list 822, 1492, 1519
\inserts@listR 866, 1331,
1334, 1344, 1358, 1359, 1488, 1508
\instanzaLfalse 1794, 2016
\instanzaLtrue 765
\instanzaRfalse 1795, 2017
\instanzaRtrue 788
\interlinepenalty 1567
\itemcount@ 1439, 1441,
1446, 1449, 1467, 1469, 1474, 1476

L

\l@d@nums 712, 715, 721, 724
\l@d@set 410, 641, 642
\l@dbbl@set@language 1660, 1683
\l@dbfnote 1483
\l@dc@maxchunks 828, 830,
872, 874, 1617, 1627, 1635, 1642
\l@dcalc@maxoftwo .....
..... 1917, 2080, 2214, 2258
\l@dcalc@minoftwo .. 1925, 2004, 2080
\l@dcalcnum 1158
\l@dchecklang 1660
\l@dchset@num 284, 287, 410
\l@dcsnotetext 1448, 1451
\l@dcsnotetext@l 1451, 1468
\l@dcsnotetext@r 1451, 1475
\l@demptyd@ta 954, 1019
\l@dend@stuff 63, 113, 128, 136
\l@dgetline@margin 172
\l@dgetsidenote@margin 1430
\l@dld@ta 982,
1245, 1257, 1270, 1282, 1291, 1303
\l@dleftbox .....
.. 935, 965, 975, 1756, 1951, 1954
\l@dlinenumR 220
\l@dlsn@te 984
\l@dmake@labels 1403
\l@dmake@labelsR 1391, 1408
\l@dminpagelines .....
.... 1886, 1926, 2005, 2102, 2116
\l@dnumpstartsL .....
..... 827, 828, 830, 832, 847,
911, 1621, 1653, 1714, 1715,
1791, 1853, 1897, 1898, 2014, 2197
\l@dnumpstartsR .....
.. 47, 871, 872, 874, 876, 890,
932, 1621, 1654, 1714, 1715,
1792, 1856, 1897, 1898, 2015, 2245
\l@doldbbl@set@language 1682
\l@doldselectlanguage 1681, 1685, 1690
\l@dpagefullfalse .....
.... 2093, 2129, 2134, 2141, 2146
\l@dpagefulltrue 2093,
2128, 2133, 2140, 2145, 2224, 2268
\l@dpagingfalse 14, 731, 746
\l@dpagingtrue 740
\l@dpairingfalse 12, 734, 745
\l@dpairingtrue 730, 739
\l@dpscL 950,
955, 968, 992, 995, 997, 1623,
1655, 1723, 1729, 1735, 1738,
1740, 1789, 1853, 1866, 1907,
1913, 1918, 1921, 1929, 1943,
1944, 1946, 2012, 2185, 2187,
2190, 2197, 2208, 2209, 2214,
2216, 2218, 2225, 2226, 2293, 2348
\l@dpscR 1015, 1020, 1033,
1624, 1656, 1724, 1730, 1746,
1749, 1751, 1790, 1856, 1869,
1908, 1914, 1922, 1930, 1974,
1975, 1977, 2013, 2233, 2235,
2238, 2245, 2254, 2255, 2258,
2260, 2262, 2266, 2267, 2301, 2350
\l@drd@ta 985,
1247, 1255, 1272, 1280, 1293, 1301
\l@drightbox .....
935, 1030, 1041, 1761, 1982, 1985
\l@drsn@te 986
\l@dsamepagefalse .....
.... 2093, 2128, 2133, 2140, 2145
\l@dsamepagetrue .....
.... 2093, 2129, 2134, 2141, 2146
\l@dsetupmaxlinecounts .. 1634, 1651
\l@dsetuprawboxes 1626, 1650
\l@dskipnumberfalse 1222
\l@dskipnumbertrue 1118
\l@dunhbox@line 985, 1000, 1003
\l@dusedbabelfalse 1658, 1678
\l@dusedbabeltrue 1658, 1680
\l@duselanguage .....
.... 1670, 1733, 1744, 1939, 1970
\l@dzeromaxlinecounts ... 1634, 1652
\l@dzeropenalties 899, 920, 1718, 1902
\l@prev@nopbR 54, 2280

\l@prev@pbR 53, 2279
\l@pscL 1623
\l@pscR 1623
\label@refs 1383, 1385, 1391, 1395, 1397, 1403
\labelref@list 1394, 1397, 1426
\labelref@listR 1377, 1382, 1385, 1422
\languagename . . 1661, 1662, 1664–1667
\last@page@num 349, 355
\last@page@numR 335
\lastbox 958, 1023
\lastskip 612, 618
\Lcolwidth *4, 5*, 16, 742, 841, 966, 975, 1741
\led@err@BadLeftRightPstarts 29, 1715, 1898
\led@err@LeftOnRightPage . . . 32, 2035
\led@err@LineationInNumbered . . . 144
\led@err@ManyLeftnotes 1469
\led@err@ManyRightnotes 1476
\led@err@ManySidenotes 1449
\led@err@NumberingNotStarted . . . 89
\led@err@numberingShouldHaveStarted 116
\led@err@NumberingStarted 44
\led@err@PendNoPstart 897, 918
\led@err@PendNotNumbered . . . 894, 915
\led@err@PstartInPstart . . . 817, 861
\led@err@PstartNotNumbered . 813, 857
\led@err@RightOnLeftPage . . . 32, 2042
\led@err@TooManyPstarts 24, 26, 829, 873
\led@mess@NotesChanged 86
\led@mess@SectionContinued 111, 126, 134
\led@nopbnumR 2284, 2285
\led@nopbR 2283, 2285
\led@pb@setting 1819, 1827, 2127, 2131, 2139, 2143, 2154, 2155, 2165, 2166
\led@pbnumR 2282, 2285
\led@pbR 2281, 2285
\led@warn@BadAction 1120
\led@warn@BadAdvancelineLine 380, 386
\led@warn@BadAdvancelineSubline 366, 372
\led@warn@BadLineation 161
\led@warn@BadSetline 631
\led@warn@BadSetlinenum 639
\led@warn@DuplicateLabel 1410
\ledgroupnotesL@false 2309
\ledgroupnotesL@true 2297
\ledgroupnotesR@false 2312
\ledgroupnotesR@true 2305
\ledllfill 985
\lednopb 784
\lednopbnum 2154, 2281
\lednopbnumR 2165, 2281
\lednopbR 784, 2283
\ledpb 783
\ledpbnum 2155
\ledpbnumR 2166, 2281
\ledpbR 783, 2281
\ledRcol@false 1044
\ledRcol@true 1013
\ledRcolfalse 15, 757, 790
\ledRcoltrue 774
\ledrlfill 985
\ledsavedprintlines *7*, 1364
\ledsectnomark 991
\ledsectnotoc 990, 1942, 1973
\ledstrutL 1951, 1954, 2020
\ledstrutR 1982, 1985, 2020
\ledthegoal 2101, 2115, 2171
\leftlinenumR 220, 1245, 1257
\leftpstartnumL 1265
\leftpstartnumR 1265
Leftside (environment) *5*, 756
\Leftsidehook 763, 768
\Leftsidehookend 767, 768
\line@list 719, 723
\line@list@stuff 127
\line@list@stuffR . . 62, 112, 135, 587
\line@listR . 81, 238, 251, 563, 710, 714
\line@margin 177, 1275
\line@marginR 170, 1250, 1296
\line@num . 352, 384, 385, 387, 405, 416, 417, 445, 570, 1077, 1424, 2202
\line@numR 55, 227, 234, 289, 323, 342, 378, 379, 381, 398, 412, 413, 436, 556, 560, 574, 1059, 1131, 1140, 1229, 1231, 1233, 1234, 1420, 2250
\lineation 166, 167, 785
\lineation* *6*, 166
\lineationR *6*, 142, 168, 785
\linenum@out 609, 615, 621, 627, 634, 642, 647, 651, 1393, 1776, 1877, 2074

\linenum@outR 584, 590, 592, 596, 597, 599, 600, 603, 604, 611, 614, 620, 626, 633, 641, 646, 650, 655, 1381, 1780, 1882, 2077, 2281–2284
\linenumberlist 1230, 1234
\linenumincrement *5*, 191
\linenumincrement* *5*, 191
\linenummargin 170
\linenumr@p 1367, 1371, 1420, 1424
\linenumrepR 217, 227
\linenumsep 222, 224, 1312, 1315, 1324, 1327
\linesinpar@listL 243, 259, 571, 2048, 2051
\linesinpar@listR 243, 255, 575, 2054, 2057
\linesonpage@listL 260, 579, 2061, 2064
\linesonpage@listR 256, 582, 2067, 2070
\list@clear 251–256, 259, 260, 262, 822, 866
\list@clearing@reg 258
\list@create 238–241, 243–245, 1331, 1377
\listcsxadd 357, 2285–2288
\lock@disp 1190, 1194, 1199
\lock@off 494, 495, 503, 650, 651
\lock@on 646, 647

M

\managestanza@modulo 1593
\maxchunks *3*, 1617
\maxlinesinpar@list 243, 262
\memorydump *7*, 762, 779
\memorydumpL 121, 762
\memorydumpR 121, 779
\message 61
\multiply 1164

N

\n@num 531, 655
\n@num@reg 537
\namebox 950, 955, 1015, 1020, 1601, 1866, 1869, 2185, 2233
\NeedsTeXFormat 2
\new@line 1000, 1003
\new@lineL 608, 985
\new@lineR 610
\newbox 795, 935, 936, 1602
\newcommandx 805, 849, 892, 913
\newcounter 95–98, 182, 184, 186, 188, 798, 799, 801, 802
\newif 5, 13, 39, 138, 139, 585, 754, 755, 1319, 1524, 1658, 1817, 1818, 1850, 1862, 2093, 2095, 2181, 2182
\newlength 1845, 1848
\newmarks 2289–2291
\newnamebox 1601, 1629, 1630
\newnamecount 1612, 1637
\newsavebox 1708, 1709
\newwrite 584, 2278
\next@absline 1820, 1822, 1824, 2132–2134
\next@abslineR 1821, 1823, 1825, 2144–2146
\next@action 279
\next@actionline 276, 278
\next@actionlineR 268, 270, 1089, 1127, 1149, 1151
\next@actionR 271, 1090, 1128, 1129, 1134, 1135, 1143, 1152
\next@insert 823
\next@insertR 867, 1335, 1338, 1340, 1343, 1347
\next@page@num 356, 428
\next@page@numR 59, 292, 294, 346, 425
\no@expands 663, 686
\noindent 847, 890, 911, 932
\normal@page@breakR 52
\normal@pars 74, 826, 870
\normalbfnoteX 1497
\notefontsetup 2335
\noteschanged@true 79, 82, 711, 720, 1337
\num@lines 900, 1719, 1903
\num@linesR 794, 921, 1720, 1904
\numberedpar@true 842, 885
\numberingRfalse 73
\numberingRtrue 49, 106, 131
\numberingtrue 123
\numberpstartfalse *8*
\numberpstarttrue *8*
\numdef 992, 1467, 1474, 1820, 1821, 2132, 2144, 2348, 2350
\numgdef 1439, 1446, 2153, 2164
\numlabfont 227
\numpagelinesL 1886, 1949, 1962, 1966, 2102, 2155, 2368

\numpagelinesR 1886, 1980, 1993, 1997, 2116, 2166

O

\old@otherlanguage 1695
\old@startstanza . . 764, 765, 787, 788
\oldchapter 737, 747
\one@line . . . 955, 958, 985, 1000, 1003
\one@lineR 794, 1020, 1023
\openout 69, 592, 597, 600, 604
\otherlanguage 1695, 1696

P

\p@pstartL 844
\p@pstartR 887
\PackageError 21
\page@action 293, 422, 550
\page@num 274, 354, 1277
\page@numR 247, 266, 344, 555, 560, 1129, 1252, 1298, 1462
\pagebreak 1834
\pagegoal 2179
\Pages *4*, 1890
pages (environment) *4*, 729
\pagetotal 2027, 2101, 2115
pairs (environment) *3*, 729
\paperwidth 999, 1002
\par@line 901, 1721, 1905
\par@lineR 794, 922, 1722, 1906
\parbox 1741, 1752
\parledgroup@ . . 957, 1022, 2289, 2315
\parledgroup@beforenotes@save 908, 929, 2375
\parledgroup@beforenotesL 2373
\parledgroup@beforenotesR 2373
\parledgroup@correction@notespacing 1953, 1984, 2362
\parledgroup@correction@notespacing@final 2220, 2264, 2343
\parledgroup@correction@notespacing@init 1967, 1998, 2338, 2346
\parledgroup@notes@endL 2188, 2191, 2219, 2308
\parledgroup@notes@endR 2236, 2239, 2263, 2311
\parledgroup@notes@startL 957, 2292, 2308
\parledgroup@notes@startR 1022, 2292, 2308
\parledgroup@notespacing@correction 2333, 2365–2367
\parledgroup@notespacing@correction@accumulated 2339, 2345, 2366
\parledgroup@notespacing@correction@modulo 2340, 2367–2369
\parledgroup@notespacing@set@correction 1894, 2333
\parledgroup@series 2290, 2294, 2295, 2302, 2303, 2318, 2319, 2325, 2326
\parledgroup@type 2291, 2294, 2295, 2302, 2303, 2316, 2323
\parledgroupnotespacing . 2332, 2335
\parledgroupseries@ 2289
\parledgrouptrue 10
\parledgrouptype@ 2289
\pausenumbering 777
\pausenumberingR 105, 777
\pend *6*, 761, 782, 818, 1580
\pendL 761, 892
\pendR 782, 862, 913
\prev@abslineverse 2153–2155, 2164–2166
\prev@nopbR 2279
\prev@pbR 2279
\prevgraf 900, 921, 1719, 1720, 1903, 1904
\print@columnseparator . . 1760, 1797
\print@eledsectionL 980, 987, 1741, 1947
\print@eledsectionR . 1048, 1752, 1978
\print@lineL 970, 980
\print@lineR 1035, 1048
\printlines 1375
\printlinesR *7*, 1364
\ProcessOptions 11
\protected@csxdef 1506, 1517
\protected@edef 843, 886
\protected@write . . . 1390, 1402, 1664
\ProvidesPackage 3
\pst@rtedLfalse 38
\pst@rtedLtrue 124, 824
\pst@rtedRfalse 40, 48, 76
\pst@rtedRtrue 109, 132, 868
\pstart *6*, 27, 31, 759, 781, 1582
\pstartL 759, 797
\pstartnumfalse 1312, 1317
\pstartnumRfalse 1324, 1329
\pstartnumRtrue 1320, 1728, 2257

\pstartnumtrue 1727, 2213
\pstartR 781, 797

R

\Rcolwidth *4,* *5*, 16, 743, 884, 1031, 1041, 1752
\read@linelist 249, 588
\rem@inder 1234, 1236–1238
\resetprevline@ 1777, 1781, 2203, 2251
\resumenumbering 778
\resumenumberingR 105, 778
\rightlinenumR 220, 1247, 1255
\rightpstartnumL 1265
\rightpstartnumR 1265
Rightside (environment) *5*, 773
\Rightsidehook 768, 786
\Rightsidehookend 768, 791
\rlap 1247, 1255, 1272, 1280, 1293, 1301
\Rlineflag *7*, 215, 227, 1367, 1371, 1412
\rule 1837

S

\savebox 1741, 1752
\sc@n@list 1235, 1237
\secdef 752
\section@num 125–127
\section@numR 36, 50, 61, 62, 68, 69, 110–112, 133–135
\select@language . . . 1662, 1664–1667
\selectlanguage 1670
\set@line 664, 687, 708
\set@line@action 286, 391, 400, 407, 430, 552
\setl@dlp@rbox 1455, 1470, 1472
\setl@drp@rbox 1457, 1465, 1477
\setline 629
\setlinenum 637
\setnamebox 832, 876, 1601
\setprintlines 1365
\shiftedpstartsfalse 7
\shiftedpstartstrue 6, 8, 9
\shiftedversesfalse 7
\shiftedversestrue 6
\showlemma 673, 696
\sidenote@margin 1432, 1435
\sidenote@marginR 1429, 1460
\sidenotecontent@ 1438, 1443, 1444, 1455, 1457, 1465, 1466, 1470, 1472, 1473, 1477
\sidenotemargin 1429
\sidenotemargin* 1429
\sidenotesep 1444
\skip 2318, 2325
\skip@lockoff 495, 503
\skipnumbering *8*, 654
\skipnumbering@reg 658
\smash 1837
\splitbotmarks 2315, 2316, 2318, 2319, 2323, 2325, 2326
\splitfirstmarks 957, 1022, 2294, 2295, 2302, 2303
\splittopskip 952, 1017
\stanza@count 1562, 1576, 1588
\stanza@hang 1564, 1596
\stanza@modulo 1562, 1591
\stanzaindentbase 1542, 1552, 1589, 1592
\startlock 645
\startstanzahook 1560
\startsub 612
\sub@action 302, 451, 551
\sub@change 60, 296, 297, 303
\sub@lock 1072
\sub@lockR 57, 311, 313, 315, 318, 469, 475, 476, 478, 479, 509, 510, 512, 1054, 1110, 1112, 1113, 1115, 1171, 1211, 1213, 1215
\sub@off 620, 621
\sub@on 614, 615
\subline@num 229, 352, 370, 371, 373, 403, 443, 1073, 1078, 1425
\subline@numR 230, 234, 319, 323, 342, 364, 365, 367, 396, 434, 557, 561, 1055, 1060, 1131, 1138, 1225, 1226, 1421
\sublinenumincrement *5*, 191
\sublinenumincrement* *5*, 191
\sublinenumr@p . 1368, 1372, 1421, 1425
\sublinenumrepR 217, 230
\sublines@false 58, 300, 1100
\sublines@true 298, 1098
\sublock@disp 1173, 1177, 1182
\symplinenum 1367
\sza@penalty 1571, 1575

T

\temp@ 992, 993
\temp@spacing 2335, 2336
\textwidth 17, 19, 742, 743
\theledlanguageL . . . 1670, 1733, 1939

\theledlanguageR . . . 1670, 1744, 1970
\thepage 609, 611, 1391, 1403
\thepstart 760, 780
\thepstartL . . . . .

eledmac 0.15). 46
\Columns: Line numbering by pstart. 64
\get@nextboxR: Change \get@nextboxL and \get@nextboxR to allow to disable line numbering (like in eledmac 0.15). 75
Pstart number can be printed in side 75
v0.12
General: New new management of hangingsymbol insertion, preventing undesirable insertions. 56
v0.2
General: Added section of babel related code 60
Fix babel problems 1
\Columns: Added \l@dchecklang and \l@duselanguage to \Columns 63
\Pages: Added \l@duselanguage to \Pages 68
v0.3
General: Added \do@lineLhook and \do@lineRhook 42
Reorganize for ledarab 1
\affixline@numR: Changed \affixline@numR to match new eledmac 46
\do@actions@nextR: Used \do@actions@fixedcode in \do@actionsR 44
\do@lineL: Added \do@lineLhook to \do@lineL 41
Simplified \do@lineL by using macros for some common code 41
\do@lineR: Changed \do@lineR similarly to \do@lineL 42
Leftside: Added hooks into Leftside environment 35
\flag@end: Removed extraneous spaces from\flag@end 31
\ifledRcol: Moved \ifl@dpairing to eledmac 14
\ifpst@rtedR: Moved \ifpst@rtedL to eledmac 15
\l@dlinenumR: Simplified \leftlinenumR and \rightlinenumR by introducing \l@dlinenumR 20
\l@dnumpstartsR: Moved \l@dnumpstartsL to eledmac . 59
\ledsavedprintlines: Simplified \printlinesR by using \setprintlines 52
\ledstrutR: Added \ledtrutL and \ledstrutR 70
\normalbfnoteX: Removed extraneous spaces from \normalbfnoteX 55
\Pages: Added \ledstrutL to \Pages 68
Added \ledstrutR to \Pages . 69
\Rightsidehookend: Added \Leftsidehook, \Leftsidehookend, \Rightsidehook and \Rightsidehookend 36
\sublinenumrepR: Added \linenumrepR and \sublinenumrepR 20
v0.3.a
General: Minor \linenummargin fix 1
v0.3.b
General: Improved parallel page balancing 1
v0.3.c
General: Compatibilty with Polyglossia 1
v0.3a
\line@marginR: Don't just set \line@marginR in \linenummargin 19
v0.3b
\Pages: Added \l@dminpagelines calculation for succeeding page pairs 70
v0.4
General: No more ledparpatch. All patches are now in the main file. 1
v0.5
General: Corrections about \section and other titles in numbered sections 1
v0.6
General: Be able to us \chapter in parallel pages. 1

v0.7
General: Option 'shiftedverses' which make there is no blank between two parallel verses with inequal length. 1
v0.8
General: Possibility to have a symbol on each hanging of verses, like in the french typography. Redefine the commande `\hangingsymbol` to define the character. 1
v0.9
General: Possibility to number `\pstart`. 8
Possibilty to number the pstart with the commands `\numberpstarttrue`. 1
`\ifledRcol`: Moved `\iflledRcol` and `\ifnumberingR` to eledmac 14
v0.9.1
General: The numbering of the pstarts restarts on each `\beginnumbering`. 1
v0.9.2
General: Debug : with `\Columns`, the hanging indentation now runs on the left columns and the hanging symbol is shown only when `\stanza` is used. 1
v0.9.3
General: `\thepstartL` and `\thepstartR` use now `\bfseries` and not `\bf`, which is deprecated and makes conflicts with memoir class. 1
v1.0
General: Compatibility with eledmac. Change name to eledpar. . 1
Debug in lineation by pstart . . 17
v1.0.1
General: Correction on `\numberonlyfirstinline` with lineation by pstart or by page. 1
v1.1
General: Shiftedverses becomes shiftedpstarts. 1
`\pstartR`: Add `\labelpstarttrue` (from eledmac). 37
v1.1.1
`\pstartR`: Correct `\pstartR` bug introduced by 1.1. 37
v1.1.2
`\affixside@noteR`: Remove spurious space between line number and line content 54
v1.12.0
General: `\beginnumbering` is defined only on eledmac, not on eledpar. 15
`\Columns`: Modify `\Columns` to enable to add section's title. . . . 62
Suppress `\l@dchecklang` from `\Columns`. 63
`\l@dchecklang`: Suppress `\l@dchecklang` which didn't work and was not logical, because both columns could have the same language but not the main language of the document. 60
`\Pages`: Modify `\Pages` to enable to add section's title. 67
v1.2
General: Support for `\led`⟨*section*⟩ commands in parallel texts. . . . 1
v1.2.1
`\initnumbering@sectcountR`: For the right section, the counter is defined only once. 16
v1.3
General: Manage RTL language. . 33
v1.3.2
General: Debug with some classes. 1
v1.3.3
General: Debugging the left notes of the right column. 54
`\l@dbfnote`: Spurious space with footnote in right column. 55
v1.3.4
General: Allow use of commands in sidenotes, as introduced by eledmac 1.0. 54
v1.3.5
`\normalbfnoteX`: Allows one to redefine `\thefootnoteX` with alph when some packages are loaded. 55

v1.4
General: Added \do@insidelineLhook and \do@insidelineRhook . . . 42
v1.4.1
\normalbfnoteX: Fix bug with normal familiar footnotes when mixing RTL and LTR text. . . 55
astanza: Enable the use of stanzaindentsrepetition within astanza environnment. 57
v1.4.2
\line@list@stuffR: Open / close immediatly the line-list file when in minipage, except if the minipage is a ledgroup. 30
v1.4.3
General: Corrects a false hanging verse when a verse is exactly the length of a line. 1
\inserthangingsymbolR: Hang verse is now not automatically flush right. 56
\pendL: Spurious spaces in \pendL. 39
\pendR: Spurious spaces in \pstartR. 40
\pstartR: Spurious spaces in \pstartL and \pstartR. 37
v1.5.0
General: Add, as in eledmac, features to manage page breaks. . . 1
\sublinenumincrement*: Add starred version of \firstlinenum, \linenumincrement, \firstsublinenum, \sublinenumincrement to change both Left and Right-side. 19
v1.6.0
General: Add tool and documentation for parallel ledgroups . . . 10
v1.7.0
General: Add, as in eledmac, features to make crossrefs with pstart numbers. 1
v1.8.0
General: \l@dlsnote,\l@drsnote and \l@dcsnote defined only one time, in eledmac. 54
Add \beforecolumnseparator and \aftercolumnseparator. . 4
Add \columnsposition. 4
Add, as in eledmac, new system of sectioning commands. 1
Add, as in eledmac, possibility to insert something after \pends / verses. 1
Add, as in eledmac, possibility to insert something between \pstarts / verse. 1
Change \do@lineR and \do@lineR to allow new sectioning commands. 41
Compatibility with musixtex. . . 1
Debug eledmac sectioning command after using \resumenumbering. 1
New sectioning commands, as in eledmac. 13
\pendL: As in eledmac, \pendL can have an optional argument. . . 39
\pendR: As in eledmac, \pendR can have an optional argument. . . 40
\print@columnseparator: Move some code of \Columns to \print@columnseparator. . . . 64
\pstartR: As in eledmac, \pendL and \pendR can have an optional argument. 37
\sidenotemargin*: \sidenotemargin is now directly defined in eledmac to be able to manage eledpar. 54
Add \sidenotemargin* 54
\theledlanguageR: Correct left/right language setting with polyglossia. 62
v1.8.1
\do@lineL: Fix a bug with critical notes a the begining of a page, (maybe added by v1.8.0) (?). . 41
\do@lineR: Fix a bug with critical notes a the begining of a page, added by v1.8.0 (?). 42
v1.8.2
General: Debug \eledxxx with some paper size 1
Debug left and side note (bugs added by 1.8.0) 1
\eledpar@error: Errors specific to eledpar send to eledpar handbook 14

`\flag@end`: `\flag@start` and `\flag@end` are now defined only one time for eledmac and eledpar 31

`\lineation*`: Add `\lineation*` . 18